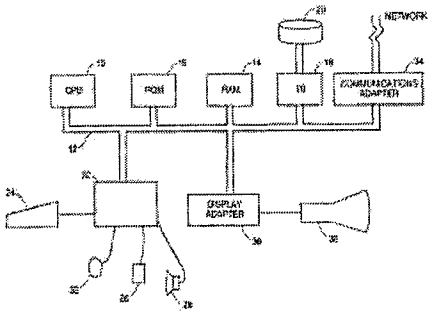
CONCURRENT FRAMEWORK SYSTEM

Publication number: JP8509825 (T)
Publication date: 1996-10-15
Inventor(s):
Applicant(s):
Classification:
- international: ***G06F13/00; G06F17/21; G06F19/00; G06F3/048; G06F3/14; G06F9/46; G06Q10/00;*** **G06F13/00; G06F17/21; G06F19/00; G06F3/048; G06F3/14; G06F9/46; G06Q10/00;** (IPC1-7): G06F13/00; G06F17/00; G06F17/21; G06F17/60; G06F3/14; G06F9/46
- European: G06Q10/00F
Application number: JP19940518956T 19940106
Priority number(s): WO1994US00341 19940106; US19930023987 19930226

Also published as:
JP3341893 (B2)
WO9419752 (A1)
US5446842 (A)
EP0672282 (A1)
EP0672282 (B1)
more >>

Abstract not available for JP 8509825 (T)
Abstract of corresponding document: **WO 9419752 (A1)**

A method and apparatus for an innovative object oriented framework system is disclosed. The system uses an innovative framework architecture to provide concurrent access to a framework application by multiple users. The users can collaborate over the application and jointly produce a finished product.



(19)日本国特許庁（ＪＰ）　(12) **公表特許公報**（Ａ）　(11)特許出願公表番号

特表平8－509825

(43)公表日　平成８年(1996)10月15日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	
Ｇ０６Ｆ　17/00		9168－5L	Ｇ０６Ｆ　15/20	Ｚ
3/14	３１０	9174－5E	3/14	３１０Ｅ
9/46	３４０	7737－5B	9/46	３４０Ａ
13/00	３５１	7368－5E	13/00	３５１Ｆ
17/21		9168－5L	15/21	Ｚ

審査請求　未請求　予備審査請求　有　（全110頁）　最終頁に続く

(21)出願番号　特願平6－518956
(86)(22)出願日　平成６年(1994)１月６日
(85)翻訳文提出日　平成６年(1994)12月27日
(86)国際出願番号　ＰＣＴ／ＵＳ９４／００３４１
(87)国際公開番号　ＷＯ９４／１９７５２　Equivalent to this literature
(87)国際公開日　平成６年(1994)９月１日
(31)優先権主張番号　０８／０２３，９８７
(32)優先日　1993年２月26日
(33)優先権主張国　米国（ＵＳ）

(71)出願人　タリジェント　インコーポレイテッド
アメリカ合衆国　95014　カリフォルニア州　クパチーノ　ノース　デ　アンザ　ブールバード　10201
(72)発明者　シェーファー，アーノルド
アメリカ合衆国　94002　カリフォルニア州　ベルモント　スカイマウント　コート　5
(72)発明者　アンダーソン，デイヴィッド　，アール.
アメリカ合衆国　95014　カリフォルニア州　クパチーノ　ウエスト　エステイツ　ドライブ　19761
(74)代理人　弁理士　谷　義一　（外１名）

最終頁に続く

(54)【発明の名称】　並行フレームワーク・システム

(57)【要約】
並行フレームワーク・システムに関するもので、革新的なオブジェクト指向フレームワーク・システムのための方法と装置が開示されている。システムは革新的なフレームワーク・アーキテクチャを使用して、複数のユーザがフレームワーク・アプリケーションを並行にアクセスすることを可能にしている。ユーザはアプリケーションで共同作業を行って、共同で完成プロダクトを作ることができる。

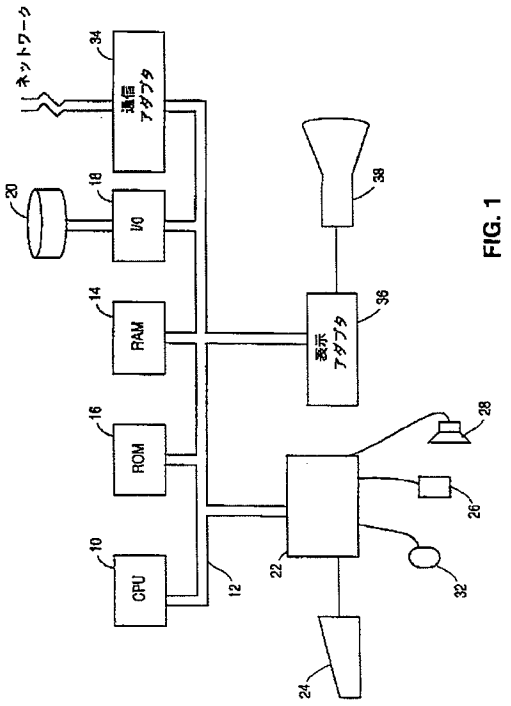


FIG. 1

【特許請求の範囲】

１．アプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置であって、

（a）第１の動的ユーザ・コマンドを解析する処理手段と、

（b）第１の動的ユーザ・コマンドを第２のユーザへ配布する処理手段と、

（c）第１の動的ユーザ・コマンドを第１と第２ユーザ・アプリケーションに適用する処理手段とを具備することを特徴とするアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

２．ひとりのユーザだけがコマンドを入力することを許可する処理手段を含むことを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

３．２以上のユーザからのコマンドを解析する処理手段を含むことを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

４．第１の動的ユーザ・コマンドはワード・プロセッシング・コマンドであることを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

５．第１の動的ユーザ・コマンドはグラフィックまたはイメージング・コマンドであることを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

６．第１の動的ユーザ・コマンドはマルチメディア、データベースまたはオペレーティング・システム・コマンドであることを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

７．第１の動的ユーザ・コマンドは反復コマンドであることを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

８．第１の動的ユーザ・コマンドは非反復コマンドであることを特徴とする請求

の範囲第１項に記載のアプ

リケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

９．無許可ユーザ・コマンドをトラップする処理手段を含むことを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

10. 第１の動的ユーザ・コマンドをユーザのシステムの特性に基づいて、異なった方法で適用する処理手段を含むことを特徴とする請求の範囲第１項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能をもつ装置。

11. アプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法であって、

(a) 第１の動的ユーザ・コマンドを解析し、

(b) 第１の動的ユーザ・コマンドを第２のユーザへ配布し、

(c) 第１の動的ユーザ・コマンドを第１と第２ユーザ・アプリケーションに適用するステップからなることを特徴とするアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

12. ひとりのユーザだけがコマンドを入力することを許可するステップを含むことを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

13. ２以上のユーザからのコマンドを解析するステップを含むことを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

14. 第１の動的ユーザ・コマンドはワード・プロセッシング・コマンドであることを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

15. 第１の動的ユーザ・コマンドはグラフィックまたはイメージング・コマンド

であることを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

16. 第１の動的ユーザ・コマンドはマルチメディア、データベースまたはオペレーティング・システム・コ

マンドであることを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

17. 第１の動的ユーザ・コマンドは反復コマンドであることを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

18. 第１の動的ユーザ・コマンドは非反復コマンドであることを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

19. 無許可ユーザ・コマンドをトラップするステップを含むことを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

20. 第１の動的ユーザ・コマンドをユーザのシステムの特性に基づいて、異なった方法で適用するステップを含むことを特徴とする請求の範囲第11項に記載のアプリケーションの少なくとも２ユーザ間で並行フレームワーク処理を行う機能を提供する方法。

【発明の詳細な説明】

並行フレームワーク・システム

著作権所有表記

関連特許出願の相互参照

本特許出願は、1992年12月23日出願の特許出願（発明の名称「オブジェクト指向フレームワーク・システム」（Object Oriented Framework System）、Debra L. Orton、David B. Goldsmith、Christoper P. MoellerおよびAndrew G. Heninger、被譲渡人Taligent）の関連出願であり、その開示内容は参照により本明細書の一部を構成するものである。

発明の分野

本発明は、一般的には、コンピュータ・ドキュメントに関し、より具体的には、フレームワークの並行使用に関する。

背景技術

ワークステーション・ソフトウェアの開発者の間で、ユーザ・インターフェイス内の一貫性を維持しながらフレキシブルなソフトウェア環境を提供することの重要性が増してきている。この種の動作環境を提供する初期の試みがヘルナンデスらの米国特許第4,686,522号に開示されている。この特許はカーソルの位置にある動的オブジェクトをユーザが引き出して、そのオブジェクトから種々の関数を引き出すことができるグラフィックとテキストの結合処理システムについて議論している。この種のユーザとの自然な相互作用あるいは対話はユーザ・インターフェイスを改善し、アプリケーションをより直感的にする。

また、オブジェクト指向アプリケーションは、どのアプリケーションが現在アクティブであるか、また多数の並行ユーザがアプリケーションをどのような使い方をしているかに関係なく、ユーザとの一貫性のある対話（interactive）イン

ターフェイスを反映している

必要がある。本件出願人が知っている公知文献には、いずれも、すべてのオブジェクト指向アプリケーションが統一的に機能することを可能にする革新的なハードウェアおよびソフトウェア・システム機能が記載されていない。

本発明が解決しようとしている問題のいくつかの解決を試みているエディタの例として、Ensembleがある（Ensemble並行オブジェクト指向グラフィックス・エディタ（Ensemble Concurrent Object-Oriented Graph ics Editor）、ACM,0-89 791-543-7（1992）を参照）。En sembleはＸ－ウィンドウをベースとしたオブジェクト指向グラフィック・エディタであり、UCLA提供のtgifグラフィックス・エディタに準拠している。EnsembleはUNIX 4.3bsdソケットに基礎を置いているので、スタンドアロン（独立型）プログラムとして使用することも、フロリダ大学の分散会議システム（Distributed Conferencing System－DCS）のアプリケーションとして使用することも可能である。これは、暗黙に置かれた書込みロック（write lock）を使用しており、このロックはオブジェクトが選択されると置かれ、オブジェクトの選択が解除されると除去される。複数のユーザはファイルを同時並行に読み取ったり、編集したりすることができ、すべてのユーザはロックが除かれると更新を受け取ることができる。ポインタは相互の合意によって共用されるので、ユーザ

は必要とする程度まで協力し合うことができる。Ensembleはロックをベースとして、オブジェクト指向グラフィック編集を解決するプロトタイプである。

発明の概要

本発明は、１つまたは２つ以上のコンピュータ・システム上で稼働している１つまたは２つ以上のアプリケーションを動的に同期化するシステムおよび方法を提供することによって、従来技術の欠点を解消している。これらのアプリケーションは一貫性のある状態でオペレーションを開始し、この一貫性はコマンドが制御システムから入力されたとき、コマンドを各アプリケーションに分散化することによって維持される。別の実施例によれば、コマンドを入力して、分散化する

２つ以上のシステムが管理される。また、アプリケーションを変更しないが、１つのシステムにだけ影響を与えるようなコマンドを１つのシステムから入力できるようにする実施例も用意されている。

図面の簡単な説明

第1A図は、本発明のパーソナル・コンピュータ・システムのブロック図である。

第1B図は、本発明による表示である。

第２図は、本発明によりアプリケーションを生成するために使用されるツールを示す。

第３図は、本発明によるコマンド・プロセスのフロー図である。

第４図は、本発明によるチェックボックス・コントロールである。

第５図は、本発明によるチェックボックス・コントロールの活性化を示す。

第６図は、本発明によるチェックボックスの更新を示す。

第７図は、本発明によるチェックボックス・コントロール処理の要約を示す。

第８図は、本発明によるコントロール・パネルを示す。

第９図は、本発明によるダイアログ・ボックスを示す。

第10図は、本発明によるダイアログ・ボックス・カラー・コントローラを示す。

第11図は、本発明によるラジオ・ボタンを示す。

第12図は、本発明によるメニュー状態処理の詳細なフローチャートを示す。

第13図は、本発明による表示の絵を示す。

第14図は、本発明によるアトミック（atomic）実行の詳細論理を示す。

第15図は、本発明によるスマート・ラベル処理と関連する詳細論理を示す。

第16図は、本発明によるスマート・ウインドゥ・ラベル処理の詳細論理を示す。

第17図は、本発明により動かされ選択されることができるオブジェクトとの一

般的対話の間にどのようにしてオブジェクトが生成されるかとどのようにしてオブジェクトが相互に通信するかを示す。

第18図は、本発明による通知ソース・オブジェクトのためのオブジェクト生成通知フローチャートである。

第19図は、本発明による適当なユーザ・インターフェイス要素を選択することと関連する詳細論理を示すフローチャートである。

第20図は、本発明によるスクロールと関連する詳細論理を示すフローチャートである。

第21A図，第21B図，第21C図は、本発明によるウインドゥ・スクロールを示す。

第22図は、本発明によるタスク管理のクラス階層を示す図である。

第23図は、TTeamHandleによってメイン・タスクを別のチームで作成するときのプロセスを示す図である。

第24図は、本発明による詳細ロジック（論理）のフローチャートである。

第25図は、アップル・マッキントッシュなどの従来

システムで使用されている代表的なトラッキング（追跡）ループを示す図である。

第26図は、本発明による抽象トラッカ・ループの例を示す図である。

第27図は、本発明によるモデル・ベースのトラッカ・ループの例を示す図である。

発明の詳細な説明

本発明は、IBM（登録商標）社のPS/2（登録商標）あるいはアップル（登録商標）社のマッキントッシュ（登録商標）コンピュータのようなパーソナル・コンピュータ上に駐在するオペレーティング・システムで実現されることが望ましい。代表的なハードウェア環境を第1A図に示す。この図は、従来のマイクロ・プロセッサのような中央演算装置10と、システム・バス12に内部接続された多数の他のユニットを有する本発明のワークステーションの典型的なハードウェア構成を示している。第1A図に示されるワークステーションは、ランダム・アクセス・メ

モリ（RAM）14と、リード・オンリ・メモリ（ROM）16と、ディスク・ユニット20のような周辺装置をバスに接続するためのI/Oアダプタ18と、キーボード24、マウス26、スピーカー28、マイクロフォン32、および／あるいはタッチ画面装置（図示せず）のような他のユーザ・インターフェイス

装置をバスに接続するためのユーザ・インターフェイス・アダプタ22と、ワークステーションをデータ処理ネットワークに接続するための通信アダプタ34と、バスを表示装置38に接続するための表示アダプタ36とを具備している。ワークステーションは、その上に駐在するIBM社のOS/2（登録商標）オペレーティング・システムあるいはアップルのシステム／７（登録商標）オペレーティング・システムのようなオペレーティング・システムを有する。

本発明は、オペレーティング・システムとエンド・ユーザ、開発者、およびシステム・ベンダのためのパーソナル・コンピュータの使用に革命をもたらすように意図された開発環境からなる新規なオブジェクト指向システム・ソフトウェア・プラットフォームである。このシステムは、完全なスタンド・アローン・タイプのネイティブ・オペレーティング・システムであり、また高性能パーソナル・コンピュータのためのバックグラウンドから達成された開発環境である。本発明は、フレームワークの価値、クラス・ライブラリおよび新世代オブジェクト指向プログラミング環境を含む完全なオブジェクト指向システムであり、サードパーティのアプリケーション・ソフトウェアの開発の経済性を基本的に改善するよう意図されている。本発明は完全にポータブルなオペレーティング・システムである。

従来のオペレーティング・システムは、ソフトウェア開発者がソフトウェアを創作するために使用できる１組のサービスを提供する。それらのプログラムはオペレーティング・システム環境全体の中に非常に緩く統合されている。例えば、DOSアプリケーションはマシン全体を支配する。これは、ユーザに関する限り、アプリケーションはオペレーティング・システムであることを意味する。マッキントッシュ（登録商標）およびウィンドウズ・オペレーティング・システムでは

、アプリケーションは同じように見え、それらはアプリケーション間での切り出し（カット）と貼り付け（ペースト）をサポートしている。この共通化によりユーザが単一環境で多数のアプリケーションを使用することが容易となった。しかしながら、その共通化はサービスとフレームワークに組み入れられていないので、ソフトウェアを開発することはまだ非常に困難である。

本発明では、“アプリケーション”を書くことは、オペレーティング・システム環境に統合されるオブジェクトの組を創造することを意味する。ソフトウェア開発者は、ソフトウェアを開発するための複雑なサービスの組とフレームワークの両方のためのオペレーティング・システムにたよることができる。本発明のフレームワークは、ソフトウェア開発者が基盤を構築するよりはむしろ問題に集中することを可能とす

る強力な抽象化を提供する。さらに、ソフトウェア開発者のための基本的抽象化は、ユーザがソフトウェアを操作するために理解しなければならない基本的な概念に非常に近い。このアーキテクチャにより、複雑なアプリケーションの開発が容易となる。

このセクションでは、本発明を採用するソフトウェアを書くため４つのステップを述べる。アプリケーションの開発を意図するユーザは一般に以下の質問に関心がある。

・何をモデル化するか

ワードプロセッサではこれは入力しようとするテキストであり、スプレッドシートではそれはセル内の値と公式である。

・データをどのように表現し提供するか

再び、ワードプロセッサでは文字が適当な改行および改ページを用いて画面上に“見るものが得るもの”（wysiwyg）形式で表示され、スプレッドシートでは表またはグラフとして表示され、構造化グラフィック・プログラム（例えばマックドロー（MacDraw））では、グラフィック・オブジェクトの組として表示される。

・何が選択可能か

ワードプロセッサアプリケーションでは、選択されるのは文字の範囲であり、構造化グラフィック・プログラムではグラフィック・オブジェクトの組である。

・この選択の上で動作するコマンドは何か

ワードプロセッサではコマンドは文字のスタイルをボールドに変更することかもしれない。構造化グラフィック・プログラムにおけるコマンドはグラフィック・オブジェクトを回転させることかもしれない。第1B図は本発明による表示を示す。表示の前面にピクチャーをもってくるためのコマンドは41に描かれている。グラフィック情報のプレゼンテーションは40に描かれている。最後に、特定のグラフィック・オブジェクト、円の選択が42に示されている。

開発者は、ユーザからたずねられる同じ４つの質問に答えなければならない。幸運にも、本発明はこれら４つの質問の各々を志向するサービスとフレームワークを提供する。答えられなければならない最初の質問は、何をモデル化するかである。ワードプロセッサ・プログラムではデータは文書を構成する文字を含む。スプレッドシートでのデータはセル内の値と公式を含む。カレンダ・プログラムではデータは時間と日付に関係するアポイントを含む。本発明は、データをモデル化するのを助ける道具を提供する。テキスト、構造化グラフィックス、サウンド、および映像を含む特定のデータ形式をモデル化するためのクラスが存在する。これらの特定のクラスに加えて、本発明は、コレクション・クラス、同一制御、リカバリ・フレームワークおよびC++言語を含む、問題のモデル化をサ

ポートする多数の他の抽象化を提供する。特定のデータ形式のためにデータ・モデルをカプセル化するクラスは、そのデータ・カプセルに含まれるデータをアクセスし修正するための特定のプロトコルと、他のデータ・カプセルを埋め込み、他のデータ・カプセルに埋め込まれるためのジェネリックなプロトコルをオーバライド（override）にするためのサポートと、データが変わるときのすべての登録されたオブジェクトに対する通知と、データのプレゼンテーションを生成するためのジェネリックなプロトコルのオーバライドとを提供する。

答えなければならない次の質問はデータをどのように提供するかである。構造

化グラフィック・プログラムではグラフィック・オブジェクトの組は一般にキャンバス上に描かれる。スプレッドシートではそれは通常、セルの表あるいはグラフである。プレゼンテーション・プログラムではスライドの組あるいはアウトラインである。本発明は、データ・カプセルに含まれるデータの“ながめ（ビュー、view）”を提供する。ビューは“ビュー（view）システム”とグラフィック・システムのセルを用いて生成される。しかしながら、サウンドあるいはビデオ・クリップを動作させることもデータのプレゼンテーションとして考えられる。

次に、何が選択されるかである？

ワードプロセッサ・プログラムでは、選択されるの

は文字の範囲であり、構造化グラフィック・プログラムではグラフィック・オブジェクトの組である。スプレッドシートではセルの範囲である。本発明は、システムがサポートする全ての基本的なデータ形式のために選択クラスを提供する。ユーザによりなされた選択を表す抽象ベース・クラスは選択されたデータの、アドレス空間から独立した指定を提供する。テキストでは、これは文字に対する一対のポインタよりはむしろ文字の数値範囲であるであろう。この限定は、他のユーザとの（リアルタイムでの）共同作業のとき、選択された範囲が他のマシーンとの間で交換されるので重要である。ベース・クラスはまたジェネリック・プロトコルをオーバライドして、この選択に対応する持続的選択を生成する。持続的選択はアンカー・オブジェクトのサブクラスであり、持続的選択が変更の編集から生き延びなければならないので、対応する短命な選択より重いと考えられる。例えば、持続的テキスト選択は、テキストがその前後に挿入されるときそれ自身を調整しなければならない。アンカーはハイパーメディアのリンク、データ・フローのリンクおよびアノテーションの実行に際して使用される。

ベース・クラスはまた、データ・カプセル内に含まれるデータを吸収し、埋め込み、送り出すためのオーバライド・ジェネリック・プロトコルを提供する。ベース・クラスはそれらを生成するために使用さ

れるユーザ・インターフェイス技術から独立である。選択は、一般的に（例えば

テキストの範囲あるいはセルをトラック（追尾）して）ユーザによる直接操作を介して生成されるが、スクリプトを介してあるいはコマンドの結果として生成されてもよい。このユーザ・インターフェイスとの直交性が非常に重要である。ベース・クラスはまたデータ・カプセルをアクセスするための特定のプロトコルを提供する。カプセル・クラスの特定のサブクラスとモデル選択クラスのサブクラスとの間には非常に強い関連性がある。

最後に、この選択された範囲について動作をすることができるコマンドは何か。ワードプロセッサのプログラムでは、コマンドは文字の選択された範囲のスタイルを変更するものでもよく、構造化グラフィック・プログラムではコマンドはグラフィック・オブジェクトを回転させるものでもよい。本発明は、多くのユーザ・インターフェイス・コマンドと共に、カット、コピー、ペースト、ハイパーメディア・リンク・スタート、リンク完了、リンクの運行、リンク上でのデータのプッシュ、リンク上でのデータのプルをするためのジェネリック・コマンドを提供すると共に、全ての組込データ形式のための多数の組込コマンド・オブジェクトを提供する。ユーザにより作られたコマンドを表すアブストラクト・ベース・クラスは、ユーザ・アクションのセマンティクス（意味）を捉える責任を

負い、コマンドがドゥ（do：実行）され、アンドゥ（undo：取り消し）され、リドゥ（redo：再実行）されることができかどうかを決定する。コマンド・オブジェクトは、コマンドがドゥされた後、コマンドをアンドゥするために必要な情報のすべてをカプセル化する責任を負う。コマンドが実行される前は、コマンド・オブジェクトは、ユーザ・アクションの非常にコンパクトな表現である。ベース・クラスは、それらを生成するために使用されるユーザ・インターフェイスの技術から独立である。コマンドはユーザによる直接の操作（例えば、グラフィック・オブジェクトの移動）を介して、あるいはメニューから一般に生成されるが、スクリプトを介して生成されることはできない。ユーザ・インターフェイスとの直交性は非常に重要である。

フレームワークの利益

本発明内の抽象化にプラグを差し込む利益は、概念的モデルを提供するよりも

大きい。フレームワークにプラグを差すことはベース・オペレーティング・システム中に構成された多くの複雑な特徴を提供する。これは、比較的小さいメソッドをコールすることによりフレームワークが多数のユーザ特徴を実行することを意味する。フレームワークのための符号化への投資がいくつかの特徴に渡って影響をもたらす。

新種のデータが操作されると、新しいデータ形式がシステムの一部となる。データのカプセルを取り扱うことができる既存のソフトウェアは修正無しに新しいデータ形式を取り扱うことができる。これは、マッキントッシュ・コンピュータ・システムのような現在のコンピュータ・システムとは異なる。例えば、スクラップ・ブック・デスク・アクセサリはある種のデータを格納することができるが、テキストあるいはクイックドロー画像成分を有するデータを表示することができるだけである。対照的に、本発明のスクラップ・ブックは、オブジェクトの形でデータを取り扱うので、あらゆる種のデータを表示する。生成された新しいデータ形式はシステム提供データ形式と全く同様に振る舞う。加えて、スクラップ・ブック内のデータは、オブジェクトがデータを編集するための標準プロトコルを提供するので、編集可能である。

スクラップ・ブックの例はデータのカプセル化の長所を際立たせる。ソフトウェアがデータのカプセル化を扱うことができるように開発されていれば、アプリケーションを、新しいデータ形式を簡単に扱うように設計することができる。新しいアプリケーションは修正無しに新しい形式のデータを表示し編集することができる。

マルチレベル・アンドゥ（Multilevel Undo）

本発明はマルチレベル・アンドゥをサポートするように設計されている。この特徴を折り込むことは、しかしながら、開発者の側に余分な努力を要求しない。システムは、生成されるすべてのコマンド・オブジェクトを単に思い出すだけである。対応するコマンド・オブジェクトが存在する限り、ユーザは、データへの特定の変更をアンドゥ、即ち取り消すことができる。システムが、コマンドを保

存し、どのコマンドをアンドゥあるいはリドゥ（再実行）すべきかを決定するので、ユーザはアンドゥの手順を折り込まない。

データのカプセル化のプロトコルの一部は、データをストリームにファイルすることと他の場所および／あるいは他の時間までデータを休ませることを行う。システムは、このプロトコルを使用してドキュメントの保存を行う。デフォルトでは、ユーザのデータ・オブジェクトは、保存されるときファイルに流される。ドキュメントがオープンされているときは、データ・オブジェクトは休まされる。システムは、データ管理フレームワークを使用してディスクに書かれたデータが一貫性をもつ状態であることを確認する。ユーザは、システムが壊れたときにデータがディスク上に保たれているように、しばしばファイルを保存しようとする。システムがすべてのコマンド・オブジェクトを保持するので、本発明はこの種の保存を必要としな

い。ドキュメントの状態は、ドキュメントの最新のディスク・バージョンから始めて、その時点からのコマンド・オブジェクトを再度与えることにより再構成することができる。信頼性のために、システムはコマンド・オブジェクトが起動される毎にそれらをディスクに自動的にログするので、システムが壊れたとしてもユーザは最後のコマンド以外失うことはない。本発明はまたドキュメントのバージョン化をサポートする。ユーザはドキュメントの現在の状態からドラフトを生成することができる。ドラフトは特定の時点におけるドキュメントの不変の“スナップショット”である（ドラフトを生成する１つの理由はコメントのために他のユーザにそれを回覧することである）。システムは新しいドラフトの生成に含まれる詳細を自動的に扱う。

共同作業

上記のように、ドキュメントは、ある過去の時の状態から始め、そのとき以降なされたコマンド・オブジェクトのシーケンスを適用することにより再構築することができる。このためユーザは故障の場合に仕事を回復することができ、リアルタイム共同作業をサポートすることができる。コマンド・オブジェクトは選択されたものについて動作し、それらの選択されたものはアドレス空間に独立であ

る。従って、選択され

たオブジェクトをネットワークを介して共同作業者に送ることができ、遠隔マシーンで使用することができる。同じことがコマンド・オブジェクトについても言える。１人の共同作業者が行ったコマンドを他の者に送り同じようにそれらの者のマシーンで実行することができる。共同作業者が同一のデータコピーから始めるならば、コピーは共同作業者が変更を行うにつれて“同期して”残る。選択の生成はコマンド・オブジェクトを使用して行うので、すべての共同作業者は同じ現在の選択をもつ。

システムは、“モデル・ペース・トラッキング”として知られている特徴を用いて各共同作業者のマシーン上でマウス・トラッキングを行う。マウスの押下を扱うように生成されたトラッカー・オブジェクトはユーザがマウスを動かすにつれて一連のインクレメンタル・コマンドを生成し実行する。これらのコマンドは共同作業者に送られて各共同作業者により実行される。各共同作業者はそれが起きるにつれてトラッキング・フィードバックをビューすることになる。システムはまた共同作業ポリシーを確立する。共同作業ポリシーは、データに変更を加えるときあるいは自由に変化するとき、ターンするように強制されるかどうかを決定する。本発明はアプリケーション開発者から責任を取り除く共同作業マシーンを取り扱う。

スクリプト

コマンド・オブジェクトのシーケンスを管理するようにシステムを設計すると、システムワイドのスクリプト道具を実行することが可能となる。コマンド・オブジェクトのシーケンスはローカル・アクションのスクリプトと等価である。スクリプト機能は、あるドキュメントに適用されるコマンド・オブジェクトを単にトラッキング（追尾）し続ける。スクリプト機能はまた、スクリプトに際して選択オブジェクトを使用する。この機能は、適用する選択を変えることによりスクリプトの注文化を提供する。コマンド・オブジェクトはそれらが特定の選択に適用できるかどうかを示すためのプロトコルを含むので、システムはユーザのスク

リプトの変更が妥当であることを確認する。

ハイパーメディアのリンク

アンカーとして知られている持続的な選択はリンク・オブジェクトにより接続されることができる。リンク・オブジェクトは、その終点を形成する２つのアンカーへの参照を有する。システムでは、リンクは双方向であり、両端は等しい能力を有する。リンクのより高いレベルの使用はリンクに方向を課すことができる。単一リンク・オブジェクトは２つの標準的特徴、即ち運行とデータ・フローとをサポートする。ユーザはリンクの一端から他端へ運行する。通常、これは指

定アンカーを含み、持続的な選択をハイライトとするドキュメントをオープンすることを含む。正確な振る舞いは宛先でのアンカー・オブジェクトにより決定される。例えば、アニメーションへのリンクはアニメーションを演ずることができる。データベース・クエリーへのリンクはクエリーを実行する。

また、リンクはデータ・フローを容易にする。リンクの一端で選択されたデータは他端に転送され、そこで選択を置き換える。ほとんどの場合、その効果は、ユーザが一端で選択をコピーし、他端へ運行するようにリンクを使用し、データを貼り付けることと同じである。システムは、リンクの一端から他端への運行（例えば、宛先ドキュメントを置き、それをオープンし、宛先アンカーをビュー（視野）内にスクロールする等）が含まれる詳細を取り扱う。同様に、システムは、リンクを横切るデータの転送の詳細を取り扱う。後者は、それが参照するデータをアクセスし、修正するための選択のプロトコルを用いて実行される。

注釈（アノテーション）

本発明はシステムワイドの注釈の道具をサポートする。この道具により著者（使用者）は見直しのためドキュメントのドラフトを分配することができる。見直し者はドキュメントにポスト・ノート（posted note）を添付することができ、実行されたとき著者にドキュ

メントを戻す。著者はポスト・ノートを調べ、それぞれについてアクションを取る（著者はドキュメント内のポスト・ノートを生成することができる）。見直し

者が著者と同じソフトウェアをもつ必要はない。代わりに、見直し者は標準注釈アプリケーションを使用することができる。このアプリケーションは著者のドラフトからデータを読み、注釈可能なプレゼンテーションを生成する（そのようなプレゼンテーションの生成は標準データ・カプセルプロトコルの一部である）。

見直し者は、ドキュメント内に選択を生成することができ、その選択にポスト・ノートをリンクすることができる。ポスト・ノートと選択の間のリンクによりシステムはそれが参照する選択に“近い”ポスト・ノートを位置決めすることができる。リンクはまた注釈構造を明白にし、システムは注釈を処理するように標準コマンドを実行することができる。ポスト・ノートの内容は、単なるテキストあるいはグラフィックではなく、システム内で実行されるデータ形式でよい。ノートの内容はデータ・カプセルを用いて実行され、ノートをオープンすることはデータの編集可能プレゼンテーションを生成することになる。

データ表現

データ表現は、モデル化するデータは何かとの質門への答えと関係する。本発明はデータのモデル化を助

ける道具を提供する。テキスト、構造化グラフィックス、サウンドおよび映像を含めて、特定のデータ形式をモデル化するためのクラスがある。これらの特定のクラスに加えて、本発明は問題をモデル化するのを助ける多数の他の抽象化、コレクション（収集）クラス、同時の制御と回復フレームワーク、およびC++言語自身を提供する。本発明の主題では、特定データ形式のためのデータ・モデルをカプセル化するクラスはカプセル化クラスのサブクラスである。

カプセル化クラス（Encapsulator Class）

開発者は、カプセル化クラスから導かれるクラスを生成することにより特定形式のデータ表現に対するコンテナを生成する。例えばグラフィック・オブジェクト、様式化されたテキスト、スプレッドシートのセルのようなシステム内の各形式のデータのために、ある形式のデータのためのコンテナとして働く異なる導かれたクラスが存在しなければならない。カプセル化クラスの各々は、それに含まれるデータをアクセスし、修正するための形式の特定プロトコルを提供する。こ

のプロトコルは、一般に、データを表示するためのプレゼンテーションにより、またデータを修正するためのコマンドにより使用される。形式特定プロトコルに加えて、カプセル化クラスは、“ブラック・ボックス”としてのデータ・カプセルの他のエイリアン（異

なる見知らぬ）形式への埋め込みをサポートするジェネリック・プロトコルを提供する。このプロトコルは導かれたクラスの中で実行され、カプセル化データのためにプレゼンテーション、編集および選択の生成をサポートしなければならない。コンテナは、エイリアン・データ形式の埋め込みをサポートするように、このジェネリック・プロトコルを理解する必要があるにすぎない。

データ表現の選択

データ形式の設計者は、特定形式のデータのための表現を設計するときに、C++オブジェクト・モデルと、選択されるべき豊富な標準クラスの組との両方をもっている。データを表現するためのユニークなクラスを設計する前に、本発明により提供されるクラスが、常に考慮されるべきである。これにより、システム内の既存のクラスに同様なあるいは同一の機能を提供する新しいクラスを生成する無駄な努力を軽減することができる。これらのうちで最も基本となるのがC++オブジェクト・モデルである。設計者は、ユーザが扱うクラスを表現するためにユーザのメンタル（精神的）モデルに近づける１または２以上のクラスを生成することができる。

本発明の基本的なクラスはデータを表現する多くの標準的な方法を提供する。収集クラスは、簡単な組か

ら辞書までランク付けして、メモリ内に関連付けられたオブジェクトを収集するための多数の方法を提供する。持続的な間違いの少ないオブジェクトの収集を提供するためのディスクベースの収集がまた使用可能である。グラフィック編集のような、２次元（2D）あるいは３次元（3D）のグラフィック・モデル化を必要とするデータ形式がまたサポートされている。多数の2Dと3Dのモデル化オブジェクトが変形され、マトリクス・クラスおよび3Dカメラと共に提供される。同様に、

本発明は全世界テキスト、審美的なタイポグラフィおよび拡張可能なスタイル（様式）機構をサポートする複雑なテキスト・データ形式を提供する。本発明はまた、サウンドおよび映像のような時間ベースのメディアのためのサポートを提供する。複雑な時間制御機構が使用可能であり、種々の形式の時間ベースメディア間の同期を提供する。

プレゼンテーション・プロトコル

カプセル化クラスは、カプセル内に含まれるデータの種々のクラスのプレゼンテーションを生成するためのプロトコルを提供する。プレゼンテーションは、寸描プレゼンテーション、“ブラウズ・オンリ”（拾い読み）プレゼンテーション、選択可能プレゼンテーションおよび編集可能プレゼンテーションを含む。プレゼンテーションのためのサイズのネゴシエーション

を行い、選ばれたサイズ内にデータを適応させるためのプロトコルがまた存在する。このプロトコルを実行するカプセル化クラスのサブクラスは、他のカプセルへのデータの埋め込みをサポートする。現在サポートされているプレゼンテーションは、以下のものを含む。

・寸描（Thumbnail）－このプレゼンテーションはユーザにカプセル内に何が含まれているかを“ピーク”することができるように（覗くことができるように）意図されている。一般にサイズは小さく、そのサイズに合うようにデータを縮小および／あるいはクリップしてもよい。

・拾い読み（Browse-only）－このプレゼンテーションによりユーザは通常のサイズでデータを見ることができるようになるが、いずれのデータも選択し修正することはできない。

・選択可能（Selectable）－このプレゼンテーションは、拾い読みプレゼンテーションにより提供される能力にデータを選択する能力を加える。データそれ自身への修正を許すことなく、注釈がデータの選択に結びつけられることができるように注釈の中で使用される。選択可能プレゼンテーションは、一般に拾い読みプレゼンテーションのサブクラスとして実行される。

・編集可能（Editable）－このプレゼンテーション

は、選択可能プレゼンテーションにより提供される能力にデータを修正する能力を加える。これは、ユーザが新たなデータを生成し既存のデータを編集することを可能とするプレゼンテーションである。現在、このプレゼンテーションはそれ自身の編集のためのウインドゥを提供する。将来的にはその場で編集を可能とするプレゼンテーションのためのサポートが加えられる。編集可能プレゼンテーションは一般に選択可能プレゼンテーションのサブクラスとして実行される。

変更通知

カプセル化クラスに含まれるデータが変更されたとき、クライアント（例えばデータの使用者）に変更を通知する必要がある。カプセルは標準通知サポートのための組込クラスにあり、カプセルがデータ表現への変更をクライアントに通知することを可能とする。クライアントは、特定の変更についての通知あるいはすべての変更についての通知のためのカプセルに接続することができる。変更が起きたとき、カプセルはすべての関連するクライアントに変更についての通知を伝搬するように、このモデルに依頼する。

データのプレゼンテーション

このセクションでは、どのようにしてシステムがデータをユーザにプレゼンテーションするかについて説明する。データが一旦システムに表現されると、適当な意味ある方法でデータをユーザにプレゼンテーションすることがユーザ・インターフェイスの役割である。ユーザ・インターフェイスはユーザとモデル化データの間のダイアログを確立する。このダイアログはユーザがデータを見、あるいは他に知覚することを可能とし、ユーザにデータを修正あるいは処理する機会を与える。このセクションはデータ表現に焦点を当てている。

ユーザ・インターフェイス

開発者は、データ・カプセルと相互作用するようにデータのプレゼンテーションを容易にするクラスを生成する。プレゼンテーションからデータ・モデルを分離することにより、本発明は同じデータの多数のプレゼンテーションを容易にする。アップル社のマッキントッシュ・ファインダのようなアプリケーションは、同一データを多数にプレゼンテーションすることを制限された形で既にサポート

している。同じ時間に同じデータを異なる観点で表示することを可能とすることはある場合には有用である。これらの異なる観点は、同じデータの４つの異なる観点を示す3DCADプログラ

ムのような同じクラスのインスタンスであってもよい。各種のプレゼンテーションのため、ユーザには、モデルを表示することができるビューと、モデルを選択し修正できるトラッキング（追尾）ツールと追尾コマンドの組を前もって書くことを要求されていた。

静的プレゼンテーション

最も簡単なプレゼンテーション形式はデータの名称である。名称はデータの内容あるいは形式を示すテキスト・ストリングである。例としては、“第４章”、“1990年連邦税収”、“すべきこと”等がある。他の簡単なプレゼンテーション形式であるアイコンは、データの小さいグラフィックプレゼンテーションである。それは通常データ形式を示す。例としては、本、レポート、金融モデル、サウンドあるいは映像の記録、図面等がある。しかしながら、また、それらは印刷しているプリンタのようなステータスを表示し、あるいは図面の縮尺図のような内容を示してもよい。最後に、寸描プレゼンテーションはモデル・データの小さいビューである。たとえば、縮尺された図、本の内容の表、縮尺された手紙、あるいは長い書類の縮尺された最初のページである。拾い読みプレゼンテーションは、ユーザが通常のサイズでデータを見ることを可能とするが、データのいずれも選択しあるいは修正することはできない。

選択可能プレゼンテーション

選択可能プレゼンテーションは、ユーザがデータを見、調べ、データから情報を抽出することを可能とする。これらのプレゼンテーションは、コンテクスト、即ち、何がデータか、データはどこか、データはいつ生成されたかを提供する。それはリスト、グリッドのような構造化手法でデータを外観としてあるいは部分的にプレゼンテーションするのを助けることができる。それはまた、データ要素間の関係、データのコンテナあるいは同胞との関係、あるいは他の依存性を表示

するのに有用である。

選択可能プレゼンテーションはまたメタデータを表示することができる。例としては、ユーザが現在処理しているデータ要素を示す現在選択がある。他の形式のメタデータはデータ要素間のハイパーメディア・リンクである。ビューはまた、データについて共同作業をしている他のユーザを示すことができる。

選択可能プレゼンテーションは、通常データ形式により非常に特定される。それらはウインドウ、ビュー、およびデータ形式を最良に反映するようにカスタマイズされたユーザ・インターフェイスオブジェクトからなる。いくつかの例は、

・サウンドの記録－コントロール・パネルは可聴プレゼンテーションを実行する。ビューは音楽のスコアとしてあるいは一連の波形としてサウンドを表す。

ビューはサンプル番号あるいは時間表示を含む。

・金融モデル－モデルは公式と他のパラメータの組として見ることができる。時間についての特定のインスタンスで、あるいはスプレッドシートのような特定の入力値で、あるいは種々のグラフィックなスタイルに、モデルからのデータを表すことができる。

・本－モデルは、内容の表、インデックス、図のリストとしてみられることができる。それは、一連のページ、一連の章、あるいは連続するテキストの流れとして見られることができる。

・映像の記録－モデルは、一連の個々のフレーム、あるいは連続するプレゼンテーションとして見られることができる。ビューは追尾マーク、フレーム数、および時間表示を含む。

・他のオブジェクトを含むコンテナーオブジェクトは名称により、形式あるいは他の属性により、一連のアイコン、１組の寸描としてアルファベット順に表示されることができる。

編集可能プレゼンテーション

編集可能プレゼンテーションは、データの修正を行うことを除いて、対話的プレゼンテーションと同様である。編集可能プレゼンテーションは、マウスあるいは他のポインタでデータの直接の処理を可能とすることによりこれを行う。また

、データがメニュー項目と

他のコントロールを介してシンボル的に処理されることを可能とする。

データ・アクセス

プレゼンテーションは、データとプレゼンテーションされるべき他の情報とを決定するためにデータ・カプセルと相互に作用する（対話する）。プレゼンテーションは要求されるデータをモデルにクエリー（照会）する。プレゼンテーションは含まれるデータのすべてあるいは一部だけをプレゼンテーションしてもよいし、あるいはデータ・カプセルのデータから導かれてもよい。

変更通知

アクティブな単一のモデルの多くのプレゼンテーションが一時に存在することができるので、データは、共同作業者を含めて多くのソースから変更され得る。モデル・データに関して、各プレゼンテーションは自身を更新し続ける責任がある。これは、モデルのすべてあるいは部分が変更するとき、通知を登録することにより達成される。プレゼンテーションと関連するデータにある変更が生じたとき、プレゼンテーションは通知を受け取り、そのビューを更新する。変更通知は以下にリストされている手法のいずれかで発生することが可能である。第１に、変更通知は実際にモデ

ル・データを変更するデータ・カプセル内のメソッドから生成される。第２に、変更通知は変更を引き起こすコマンドから発生される。前に述べたように、これら２つのアプローチには利益がある。データ・カプセル内から通知を発生することはデータが変化するときにはいつでもクライアントは通知されることを保証する。コマンドからの通知の発生は、“高級”通知を可能とし、複雑な変更により生成される通知の混乱を減少させる。

通知フレームワーク外観

通知フレームワークは、オブジェクト間の変更情報を伝搬するための機構を提供する。フレームワークは、オブジェクトが、それらが依存するオブジェクトに関係することを表し、オブジェクトの変更についての通知を受け取ることを可能

とする。標準インターフェイスはクライアントに通知を提供するクラスのために提供される。通知者クラスはクライアントのリストを管理しそれらのクライアントに通知をデスパッチする手段を通知ソース・オブジェクトに提供する。通知者オブジェクトは通知を受け取るオブジェクトのクラスについての特別の知識を要求しない。接続オブジェクトは通知者オブジェクトからの通知のデスパッチを特定の通知受信オブジェクトに提供する。これらの接続オブジェクトは通知が受信者の異なるクラスに

どのようにして配信されるかを特定することを可能とする。最後に、通知オブジェクトは変更について記述した情報を輸送し、インタレストは通知ソース・オブジェクトを記述する。

通知伝搬フローチャート

第18図は通知ソース・オブジェクトに対するオブジェクト発生識別フローチャートである。処理はターミナル1800で始まり、機能ブロック1810まですぐに通過する。そこでは、通知受信者オブジェクトがそれ自身への接続を生成する。その後、機能ブロック1820で、通知受信者オブジェクトは１またはそれ以上の通知ソース・オブジェクトからの１またそれ以上の通知に対して適切なインタレストを加える。これらのインタレストは通知ソース・オブジェクトにより定義される。

クライアントオブジェクトは、機能ブロック1830、接続の際にインタレストにより特定される通知のために通知ソースに接続するよう接続オブジェクトに依頼する。その後機能ブロック1840で、接続に際しての各インタレストでは、接続はインタレスト内の通知者との通知と関連するとして登録される。次に機能ブロック1845で、システムは変更が検出されるまで待ち状態に入る。システム変更が起きたとき、コントロールは直ちに1850まで通過し、そこで通知ソース・オブジェ

クトは変化し、コールがその変更を記述する通知つきの通知者に通知する。

通知に関係する通知者と共に登録された各接続では、機能ブロック1860で、接続は通知をデスパッチするよう依頼される。次に、機能ブロック1870では、接続は通知受信者の適切なメソッドに通知をデスパッチする。最後に、機能ブロック

1880で、通知受信者は通知のための適切なアクションを起こし、テストが決定ブロック1885で行われ、他の接続が通知に関係する通知者つきで登録されているか否かを決定する。他の接続が存在するときは、コントロールは1850に進む。サービスすべき他の接続が無いときは、コントロールは機能ブロック1845に進み、次の変更を待つ。

データ仕様

データ仕様はデータ処理の選択を発行することを目的とする。ユーザが表現に含まれるデータを処理しなければならないとき、データはそのデータのサブセットを特定できなければならない。ユーザは一般にこの仕様を“選択”と呼び、システムはすべての選択クラスが降りたベース・クラスを提供する。本発明はまたシステムがサポートする基本的データ形式のすべてのための選択クラスを提供する。

モデル選択

表現内のデータのサブセットの仕様を含むオブジェクトはモデル選択クラスである。テキスト表現の場合には、ある可能な選択仕様は一対の文字オフセットである。構造化グラフィック・モデルでは、各形状にはユニークなIDが割り当てられなければならず、選択仕様は１組のユニークなIDである。仕様のどれもが選択データを直接ポイントせず、それらをデータの多数のコピーに適用することができる。

特定データのアクセス

選択は、データをアクセスし修正する表現プロトコルを理解し、ローカル・アドレス空間でデータを見つける方法を知る。コマンド・オブジェクトはデータ選択を介して表現データをアクセスし、従って仕様からローカル・モデル内のリアル・データに変換する知識を要求しない。それは選択オブジェクトの仕事であり、アドレス空間独立仕様からリアル・データへのアクセスを提供する。テキスト・カプセルでは処理はある範囲内に含まれる実際の文字に対するカプセルをクエリーすることを要求できる。グラフィック・エディタのようなベース・モデルでは、選択はリアル・オブジェクトの代用物を保持する。カプセルはリアル・オブ

ジェクトへ代用物を変換するためのルックアップ道具を提供する。

標準編集プロトコル

モデル選択クラスは選択間のデータの交換のためのプロトコルを提供する。形式ネゴシエーション、データの吸収、埋め込み、および運び出しを行うためのプロトコルを実行することにより、導かれたクラスは標準編集コマンドのほとんどのサポートを提供する。これは、システムにより提供される（カット、コピー、ペースト、データをプッシュ等の）編集コマンドが、表現されたデータ形式のために機能し、各アプリケーションの再実行を要求しないということを意味する。モデル選択クラスは、また、アンカーとリンクの交換のために直接にサポートを提供するが、表現データの交換をサポートするためいくつかのキーメソッドの導かれたクラスの実行上にある。

CopyDataは、導かれたクラスにより実行されなければならず、特定データのコピーを運び出す。実行は特定データのコピーを含む要求され形式の新たなデータ・カプセルを生成してリターンする。

AdoptDataは、仕様に関連する表現にデータを吸収しあるいは埋め込むことをサポートするように導かれたクラスにより実行されなければならない。データが吸収されるべきならば、それは受信者の表現に直接組み込まれることができる形式でなければならない。吸収されたデータは仕様で定義されたように表現に加えられる。多くのデータ形式に対して現在特定されてい

るデータを新たに吸収しデータで置き換えることは共通である。いずれかの置換されたデータはアンドゥをサポートするようにデータ・カプセルに戻される。データが埋め込まれるべきならば、カプセルはブラック・ボックスとして組み込まれ、表現の子として加えられる。

ClearDataは、関連する表現から特定されたデータをデリートするように導かれたクラスにより実行されなければならない。デリートされたデータを含む表現の元来の形式のカプセルは戻されなければならない。

ユーザ・インターフェイス

仕様を生成するユーザ・インターフェイスは一般にはデータの表現の応答性である。多数の機構がデータ形式とプレゼンテーション・スタイルに基づいて使用可能である。選択を生成するための最も好ましいユーザ・インターフェイスは直接実行である。簡単なグラフィック・モデルでは、マウスでオブジェクトを直接クリックしあるいはマウス・トラッカーを用いていくつかのオブジェクトを横切って選択ボックスをドラッグすることによりオブジェクトを選択することができる。テキストでは、選択はファインド・コマンドの結果として生成することができる。選択が生成される他の共通の手法は、“ファインド”のようなメニューコマンドの結果としてである。コマンドが発行されたあと、ドキュメントは適切な場所にスクロールされ、サーチされたテキストが選択される。

最後に、選択はスクリプトから生成され（あるいはプログラム的に発生され）る。その結果はユーザが直接選択を生成したかのようである。スクリプトのための“名称付け”選択は選択を記述するための言語を生成することを含む。例えば、テキストでは、選択は“２ページの第４節の２番目の語”である。本発明のアーキテクチャはスクリプトのためのサポートを提供する。

データの修正

データ修正（Data Modeification）は質問を発する。：この選択に関して動作することができるコマンドは何か？

ユーザが表現内に含まれるデータを修正すべきとき、システムはなされるべき修正の形式を正確に特定できなければならない。例えば、ワードプロセッサ・プログラムでは、ユーザは選択された範囲の文字のスタイルを変更することを望むことができる。あるいは、構造化グラフィック・プログラムでは、ユーザはグラフィック・オブジェクトの回転を望むことができる。データ・カプセルに含まれるデータを修正するすべてのユーザのアクションは、“コマンド・オブジェクト”により表現される。

モデル・コマンド・オブジェクト

ユーザによりなされたコマンドを表すアブストラクト・ベース・クラスはモデ

ル・コマンド・オブジェクトである。モデル・コマンド・オブジェクトのサブクラスはドゥ、アンドゥ、リドゥされることができるというようなユーザ・アクションの意味を捉える。これらのサブクラスはそれらを生成するために使用されるユーザ・インターフェイス技術から独立している。MacAPPと異なり、ユーザ・アクションの意味が知られるとすぐに、デバイス・イベントがシステムによりコマンド・オブジェクトに翻訳される。

HandleDo（HandleDo）、

ハンドル・アンドゥ（HandleUndo）、

ハンドル・リドゥ（HandleRedo）

新しいクラスのコマンドを生成することは多数のメソッドをオーバライド（override）することを含む。オーバライドするための最も重要な３つのメソッドは、Handledo, HandleUndo、およびHandleRedoである。HandleDoメソッドは、そのままであるコマンド形式とそのコマンドが適用される選択とに基づいて適切にデータ・カプセルを変更するように応答可能である。例えば、ワードプロセッサにおいてコマンドがあ

る範囲の文字に対するスタイル変更を含むならば、HandleDoメソッドはこのコマンドのあとに“アンドゥ（Undo）”する必要のある情報のすべてを保存する。スタイルの変更の例では、アンドゥ（undo）情報の保存は文字範囲の旧スタイルを記録することも含む。ほとんどのコマンドに対するアンドゥ情報は保存が非常に簡単である。しかしながら、ファインドおよび変更のようないくらかのコマンドは大量の情報を記録したあとでコマンドをアンドゥすることを含む。最後に、HandleDoメソッドは、データ・カプセルになされる変更を記述する変更通知を発行する。

HandleUndoメソッドはコマンドが“ドゥ”される（done）前にそうであった状態にドキュメントを戻す。適用されなければならないステップは上記のHandleDoでなされたステップと同様である。HandleRedoメソッドは、ドゥ（do）され、そしてアンドゥ（undo）されたあとコマンドを“リドゥ（redo）”する。ユーザはしばしばアンドゥ／リドゥ（undo/redo）の組み合わせを用いてコマンドの結果

を比較して、ドキュメントの２つの状態の間をいったりきたりする。一般に、HandleRedoメソッドは、Redoメソッドにおいて、最後に導かれた情報がこのコマンドが完了したとき再使用されることがきることを除いてHandleDoメソッドと非常に近い（情報は同じであることが保証されるので、再計算する必要はない）。

ユーザ・インターフェイス

コマンド・オブジェクトは、ユーザ・アクションの意味を捉える。実際、コマンドは（種々のユーザ・インターフェイス技術を用いて）ユーザにより最も多く生成されるが、同様に他の手法で生成することができる“ワーク・リクエスト”を表す。重要な概念は、コマンド・オブジェクトがデータ・カプセルに含まれるデータを修正するための唯一の手段を表すことである。データ・カプセルへのすべての変更は、無限のアンドゥの利益、保存無しのモデル、本発明の他の特徴が実現されるべきならば、コマンド・オブジェクトにより処理されなければならない。

コマンドを発行する最も好ましいユーザ・インターフェイスは、ある種の直接操作を含む。デバイス・イベントをコマンドに翻訳し、ユーザ・フィードバック・プロセスを“ドライブ”するように応答可能なオブジェクトはトラッカーとして知られている。本発明は組込データ形式を処理するための“トラッキング・コマンド”の豊かな組を提供する。例えば、直線、曲線、多角形等のような“ピンク（Pink）”において2Dオブジェクトのすべてを回転させ、スケーリングして、動かすためのトラッキング・コマンドがある。

コマンドを発行するための共通のユーザ・インターフェイスはメニュー・システムからあるいはコントロールを介している。メニューは生成され関連するコ

マンドの組がメニューに加えられる。ユーザがメニュー内の項目を選ぶとき、適切なコマンドが“クローン（複製）”され、コマンドのドゥ・メソッドがコールされる。プログラマはデバイス・イベントに巻き込まれない。さらに、コマンドはそれらが適用されることができる選択の形式が何かを知っているので、メニュー項目はそれらが適切でないとき自動的に薄暗くさせられる。

最後に、コマンドはスクリプトから発行することができ（あるいはプログラム的に発行することができ）、その結果はユーザがコマンドを直接発行したときと同様である。“ピンク”アーキテクチャはスクリプトするためのサポートを提供する。しかしながら、この時点ではこれらのスクリプトを生成するために使用可能なユーザ・インターフェイスはない。

組込コマンド

本発明は、多くのユーザ・インターフェイス・コマンドと同様に、カット、コピー、ペースト、ハイパーメディア・リンクのスタート、リンクの完了、リンクの運行、リンクに関するデータのプッシュ、リンクに関するデータのプルのためのジェネリック・コマンドを提供することに加えて、全ての組込データ形式のための多数の組込コマンド・オブジェクトを提供する。

他の特徴

このドキュメントの前のセクションは、本発明の基本的な特徴に焦点を当てた。本発明には、さらに進んだ特徴を実行する多くの追加される道具がある。特に、これらの道具は、モデルベースのトラッキング（追尾）、ファイリング、アンカー、および共同作業を含む。

モデル・ベース・トラッキング

トラッキング（追尾）は、直接処理ユーザ・インターフェイスの心臓部である。トラッキングにより、ユーザは、テキストの範囲を選択し、オブジェクトをドラッグし、オブジェクトを再サイズ調整し、オブジェクトを描くことが可能となる。本発明は、モデルを実際に修正することにより多数のビュー（表示）、および多数のマシーンに渡る機能にトラッキングを拡張する。トラッカーは、モデルにコマンドを発行し、モデルはすべての関係するビューへ変更通知を発行する。

モデル・ベース・トラッキングは、ドキュメント内でトラッキングするための最良の解決策であるが、それは、以下の欠点を有する。（1）モデル・ビューを、変更イベントへの急速応答を提供するように最適化しなければならない。（2）モデルは中間トラック状態を表すことができなければならない。

アンカー（Anchors）

持続性選択即ち“アンカー”はデータの仕様を表現する点で選択と非常によく似ている。違いは、アンカーはデータへの変更に渡って維持されるので、アンカーは編集変更を生き続けなければならない点である。ドキュメントの初期で述べたグラフィック選択の実行は持続的である。しかしながら、テキスト選択の実行はない。ユーザが選択の前にテキストを挿入しあるいはデリートするならば、文字オフセットを調整しなければならない。テキストアンカーを実行するためには２つのアプローチがある。第１に、テキスト表現は、スタイルが維持される手法と同様に、テキスト内を指すマーカーの集まりを維持する。アンカーはマーカーを参照するユニークなIDを含む。テキストが変更されたとき、マーカーはユニークにされるが、アンカーは同じままである。他のアプローチは、テキストに対する編集履歴を維持することである。アンカーは時間スタンプと同様に１対の文字位置を含むことができる。テキストが編集される毎に、履歴は更新され、変更（例えば、時間Ｔに位置Ｘからデリートされた５文字）を記録する。アンカーが使用されるとき、システムは、最後に使用されたとき以来起きた編集変更に基づいて文字位置を補正しなければならない。適切な時に、履歴は凝縮され、アンカーは永久的に更新される。

システムは、アンカー道具を介して“フリー”のための非常に多数の特徴を提供する。ハイパーメディア・コマンドのすべて（CreateLink、PushData、PullDataおよびFollow）は、それらの実行時にアンカーを使用する。システム中の注釈道具実行時にアンカーを使用する。ベース・データ・カプセルはアンカーとリンクの追尾を保つサービスを提供する。しかしながら、ユーザは、アンカーをプレゼンテーションを介してユーザに見えるようにする。アプリケーションはまたユーザがアンカーを選択するとき適当なコマンド・オブジェクトを発行しなければならない。アンカーとリンクのためのユーザ・インターフェイスが釘付けにされたあと、ドキュメント・フレームワークは処理を簡略化するよう付加的サポートを提供する。

ファイリング

ファイリングはデータを永久格納に保存しあるいは永久格納から再生するプロセスである。ユーザがファイリング作業をするためにしなければならないことは、データ・カプセルのための一連の操作を実行することのみである。本発明のデフォルト・ファイリングは“イメージ”ベースである。ユーザがドキュメントを開くとき、ドキュメントの内容全体はメモリに読み出される。ユーザがドキュメントを閉じるとき、ドキュメントの内容の全体がディスクに書き込まれる。

このアプローチは、簡単で、フレキシブルで、理解しやすいので採用された。異なるフォーマットでデータを格納するためには、前もって存在する標準のファイル・フォーマットとの互換性を保つために、２つのアプローチが可能である。第１に、カプセル・クラスは実際のデータへの参照を一連の流れとすることができ、実際のデータを見つけるために参照を使用することができ、あるいは新しいサブクラスがファイル・サブクラスを生成し、リターンするように定義することができる。

第１のアプローチの長所はデータ・カプセル型のドキュメント内でカプセル化することができることである。第２のアプローチの長所は、完全なドキュメントのための既存のファイル・フォーマットに正確に適合するようにした完全な自由である。

共同作業

同時ネットワーク共同作業は、２人あるいはそれ以上の人が同時に同じドキュメントを編集することを意味する。システムはまた共同作業ポリシーを確立する。即ち、ユーザがデータを変更するときあるいは自由に変更させるとき後退させられるかどうかである。開発者は共同作業ポリシーあるいは共同作業の機構を心配する必要はない。

共同作業選択スタイルのサポート

混乱の減少を助けモデル選択を増強するため、ドキュメント・アーキテクチャは共同作業者のイニシャルと望ましいハイライトの束について情報を含む共同作用者クラスを提供する。

多数選択のサポート

多数の選択をサポートするために、ユーザは、各共同作業者が選択を有するのでプレゼンテーション・ビューを修正しなければならない。アクティブな共同作業者の選択が変わるとき、標準変更通知が送られる。受動的共同作業者の選択が変わるとき異なる通知イベントが送られる。ビューは両方のイベントのために登録すべきである。どちらかのイベントに応答して取られたアクションも通常同じなので、両方のイベントのために同じハンドラー・メソッドを登録することにより経済性が実現される。

本発明によるユーザ・インターフェイス

本発明のこの部分は、基本的に、先に議論されたオペレーティング・システム・フレームワークの基本上に構築するユーザ・インターフェイスの観点に焦点が絞られる。ユーザ・インターフェイスの第１の観点はユーザがコントロールとして参照する種々のオブジェクトあるいはデータとの対話（相互作用）を管理することを可能とする機構である。

コントロール

ユーザが他のオブジェクトあるいはデータを処理するよう対話するオブジェクトはコントロールと呼ぶ。コントロールはオブジェクトあるいはデータの現在の状態を決定するためのコマンドを使用する。ユーザとの適切な対話に続いて、コントロールはコマンド・パラメータを更新し、それが実行されるようにする。コントロールの例はメニュー、ボタン、チェックボックス、およびらラジオ・ボタンである。

コントロールはオブジェクトあるいはデータの現在の状態を決定するためにコマンドを使用する。ユーザとの適切な対話に続いて、コントロールはコマンド・パラメータを更新し、それが実行されるようにする。例えば、チェックボックスはコマンド・パラメータをオンにあるいはオフにセットし、データ値を変更するためにコマンドを実行する。

多くのコントロールはそれらが処理をするデータの現在の値を表示する。例えば、チェックボックスはブール（代数）データ値が真（TRUE）のときだけチェッ

クを表示する。データが変わるにつれてコントロールの出現はここで述べられた通知システムを用いて日付を付けるよう保たれる。そのプロセスはメニュー項目をイネーブル／ディスエーブルするように使用される

プロセスと同様である。

コントロールが生成されるとき、コマンドが特定される。コントロールはこのコマンドのコピーを作り、それをフィールドfCommandに格納する。コマンドがいずれかのデータ値を供給するならばコマンドの適切なゲット（Get）とセット（Set）のメソッドのポインタがまた特定される。コントロールはこれらのメソッド・ポインタをフィールドfGetMethodとfSetMethodにそれぞれ格納する。その後、コントロールはデータ値が日付以外であることを示す通知に接続する。各コマンドはこのためにコネクトデータ（ConnectData）と呼ばれるメソッドを提供する。

各コントロールは通知を受信すべきオブジェクトとメソッドを示すfDataConnectionと呼ばれる接続オブジェクトを含む。この接続オブジェクトはコマンドへのアーギュメントとして渡される。コマンド・オブジェクトは接続オブジェクトの接続メソッドをコールして、そのデータ値に影響を与えるかも知れない各通知者とインタレストを加える。完了時に、コントロールは接続オブジェクトの接続メソッドをコールして第３図に示される接続を確立する。コントロールはそのデータ値をそのコマンドから更新する。それはコマンド（fCommand->（*fGetMethod( )）のゲット（Get）メソッドをコールすることにより行う。第５図に示されるように、コントロールはこの値を適切なフィールドに格

納する（たとえば、チェックボックスがfCheckedと名前が付けられたブール・フィールドに格納する）。その後、コントロールはその出現を更新する。画面の一部が更新を必要としていることを示して、ビュー・システムの不当化メソッドをコールすることによりこのアクションを行う。

最後に、データの変更と通知が送られる。ある点で、コントロールにより反映されているデータの値を変更するコマンドが実行される。このコマンドは、コン

トロール、メニュー項目から、あるいは直接操作により実行することができる。コントロールは第４図に示される通知を受け付けて次のユーザ選択を待つように渡される。

コントロール・パネル

コントロールのある集まりは、コントロール・パネルと呼ばれる。コントロール・パネル内のコントロールは一般に実際のデータに作用する（これはディフォルトであり、必要はない）。それらのアクションは通常直接的であり、互いに独立である。コントロール・パネルは必要によりコントロールの間で入力フォーカスの進行を制御する。コントロール・パネルはシステムですべてのユーザ・インターフェイスに渡って共有されるであろう。

ダイアログ・ボックス

コントロールの他の集まりはダイアログ・ボックスと呼ばれる。ダイアログ・ボックス内のコントロールは一般にプロトタイプのデータに作用する（これはディフォルトであり、必要はない）。それらのアクションは通常グループ内に一緒に集められ、その後ユーザがApplyボタンを押すとき一緒に実行される。ダイアログ・ボックスは必要によりコントロールの間で入力フォーカスの進行を制御する。

アクションにおけるコントロール

３つのアクションの役割を提供して、アクションにおけるコントロールを説明する。第２図は種々のコントロールを示す。役割の例はコントロール（この場合チェックボックス）、コマンド、選択およびデータ・カプセルと似た手法により使用される。

チェックボックス200　チェックボックスの役割はデータ・カプセル内に格納されたブール値を表示して変更を容易にすることである。この値はチェックの存在不存在により表現される。

コマンド210　コマンドの役割はデータ・カプセルから値を得てチェックボックスからの方向に変更することである。

選択220　選択の役割はコマンドとデータとの間をインターフェイスすること

である。

データ230　データはアクションの目標として採用される。

あなたを知ること

誰でもが第３図に示されるようによりよく互いを知りたい。コマンド310はコントロールが興味があるデータがどの通知を送ることができるかをチェックボックス300に知らせる（コマンド310がどのようにして知るかは他の者に関係がない）。チェックボックス300は次に通知のためにデータ320に接続する。

誰にも知られず、ディレクタはチェックボックス300にコマンド310と対話する最良の手法を話した。特に、それはコマンドのゲット値メソッドとセット値メソッドについて話された。チェックボックスはこの長所を少しあとで取り上げる。

データを反映すること

何かがデータに起きる－それは第４図に示されるように通知を送る。チェックボックス400はインタレストを表現したそれらについて聞く。第４図では、データからの通知はチェックボックス内にＸを置くことにより反映される情報を太くする（ボールドにする）ように表現する。

チェックボックス510はデータから通知を受信してチェックボックス510を正しく表示する処理が第５図

に示される。それは、知ろうとするコマンド520のゲット値メソッドを用いて行う。正しい値は何かをチェックボックス510に知らせる前に、コマンド520は選択を介してデータに行き、実際に正しい値を知っていることを確認する。チェックボックス510は必要によりそれ自身を更新する。

データを変更すること

ユーザはこのシーンで、第６図に示されるようにチェックボックス600を一度押す。チェックボックス600はコマンド610のセット値メソッドを用いて選択を介してデータ620の値をセットする。全体のプロセスを第７図に再び示す。

アクション時のコントロール・パネル

コントロール・パネルは第８図に示されるように一セットのコントロールを含

む単なるウインドゥである。これらのコントロールは現在の選択に作用するコマンドを含む。コントロールはコマンドがアクティブならばイネーブルされる。ユーザとの適切な対話に続いて、コントロールはコマンドを実行し、データを変更する。

サウンド・コントロール・パネル

コントロール・パネルの例として第８図に示すサウンド・コントローラを考える。このコントロール・パネルはサウンドの再生を制御する４つのボタン800, 802, 804および806を含む。各ボタンは上記の“アクション時のコントロール”のセクションに述べられたように実行する。

プレイ800　このコントロールはTPayコマンドである。このコマンドはある条件の下でのみアクティブで、それらの条件の下でのみコントロールをイネーブルにする。第１に、サウンドは適切なデータ・カプセル内で選択される。次にそれは既に再生中であってはならない。最後に、現在のサウンド位置は最後の前のどこかになければならない。押されたときPlayボタンはTPlayコマンドを実行して選択されたサウンドをスピーカーから出力させる。

ステップ802　このコントロールもまた、TPlayコマンドを含む。これは何であろう　さて、これを構成しているとき以来、TPlayコマンドがそれを再生すべき期間を示すパラメータを取ることができる。ステップボタン（Stepボタン）のために、それは単一サンプルに設定される。StepボタンはTPlayボタンに対して述べたのと同じ条件の下でのみイネーブルとされる。Stepボタンが押されると、TPlayコマンドを実行して選択されたサウンドをスピーカーから出力させる。

ストップ804　このコントロールはTStopコマンドを含む。Stopコマンドは選択されたサウンドが現在再生中であるときのみイネーブルとされる。押されたとき、StopボタンはTStopコマンドを実行して選択されたサウンドの再生を停止させ、現在のサウンド位置を開始点に設定する。

ポーズ806　このコントロールはまたTStopコマンドを含む。しかしながらStopボタンと異なり、このTStopコマンドはサウンドを開始点に巻き戻さないよう設

定される。Playボタンあるいは Stepボタンを押すと再生が中断された場所から続く。

アクション時のダイアログ・ボックス

ダイアログ・ボックスは、それが一セットのコントロールを含む単純なウインドゥであるという点でコントロール・パネルと同様である。しかしながら、選択されたデータへ作用するコントロールに変わって、それらは他のコマンドのパラメータに作用する。Applyボタンが押されるまでだけが修正されたリアル・データである。

カラー・エディタ

ダイアログ・ボックスの例として第９図に示すカラー・エディタを考える。それは３つのスライダを含む。一つはカラーの赤900、一つは青910、および一つは緑920の成分のためのスライダである。スライダ

を望みの値に調整したあと、ユーザはApply930を押して選択した範囲のカラーを変更する。

赤900，緑910，青910　ユーザにとって、これらのスライダは、ラベルを除いて同一である。すべてのコントロールと同様に、各スライダは、ユーザとの対話に続いて実行されるコマンドを含む。多くのコントロールとは異なり、特に選択されたデータにすぐに影響を与えるコントロール・パネル内のそれらと異なり、これらのスライダに含まれるコマンドは他のコマンドのパラメータ値を表示し修正する。この場合、Applyボタン内に含まれるコマンドの赤、緑あるいは青のパラメータの１つである。

Apply930　Applyボタンは、実行されるときに選択された範囲のカラーを変更するTSetColorコマンドを含む。それは３つのパラメータを有する。カラーの赤、緑および青の成分の各々に対して１つのパラメータである。これらのパラメータは表示され、ユーザとの対話に応答してスライダにより設定される。Applyボタンが押されると、このコマンドが実行され新しいカラーが設定される。カラー・エディタに伴う内部アクションを例として第10図に示す。赤1000，緑1010，青1020のスライダはTFloatControlCommandを含む。これらのコマンドはコントロー

ルが表示する単一の浮動小数点値を含む。ユーザがスライダを調整するにつれて、スライダはこの値を更新しそのコマンドを実行す

る。

TFloatControlCommandの選択によりApply1040ボタン内のTSetColorコマンドが特定される。そのパラメータの１つは各TFloatControlコマンドが実行されるときに設定される。最後に、ユーザがApply1040ボタンを押すとき、TSetColorコマンドが実行され、選択されたカラー1050が変更される。

クラス

以下のセクションはコントロールおよびダイアログ領域のクラスとそれらの基本的なメソッドを説明する。

コントロール

コントロールは１またはそれ以上のコマンドへのユーザ・インターフェイスである。コントロールはその名称通りにコマンドについての情報を表示し、また、それが現在のコンテキストの中でアクティブか否かを表示する。適切なユーザとの対話に続いて、コントロールはコマンドを実行する。適切なとき、コントロールはコマンドが修正するデータの現在値を得て、それをユーザに表示する。コマンドを実行する前にこのデータの新しい値を示すコマンド・パラメータをセットすることができる。

オプションとしてのコントロール内のコマンドの付

加的仕様でコントロールについての選択を生成するメソッド。ルックアップ・コマンドは、それらがどのくらい多くのコマンドを得て、それらがどのように格納されるかについての、サブクラスにフレキシビリティを与えるための、純粋な仮想関数である。

プレゼンテーションがオープンされクローズされるときコールされるメソッド。プレゼンテーションがオープンされるとき、コントロールはその状態に影響を与えることができる通知を接続する。プレゼンテーションがクローズされるときこれらの接続は切断される。

プレゼンテーションが活性化されまた非活性化されるときコールされるメソッド。プレゼンテーションが活性化されるとき、いくつかのコントロールはアクティブのときだけ妥当な通知に接続される。プレゼンテーションの非活性化はこれらの接続を切断する。

コントロールがイネーブルとされるかどうかに影響を与える通知者に接続しそれから切断するためにコントロールが使用するメソッド。ConnectEnabledNotifiersはコントロールがオープンされたときコマンドにより特定される通知者に接続する。DisconnectEnabledNotifiersはコントロールがクローズされるときこれらの接続を切断する。

コントロールのデータ値のプレゼンテーションに何かが影響を与えることを示す通知を受信するメソッ

ド。このメソッドはディフォルトにより何もしない。

通知のためのメソッド。クリエートインタレスト（Create interest）はコントロールのインスタンスにより特定されるインタレスト（interest）を生成する。Notifyは通知を送り、インタレストを含み込むようにオーバーロード（overload）される。

コントロールのインタレスト

単一の通知者はコントロールの多くのサブクラスの中で共有される。特定のコントロールインスタンス内のインタレストを表すために、インタレストは特定されなければならない。コントロールインタレストは特定のコントロールを指すポインタを含むインタレストである。このクラスはサブクラス無しで、通常そのまま使用される内部クラスである。

コントロールの通知

各通知者はコントロールの多くのサブクラスの中で共有されている。コントロールがどこに通知を送るかを区別するために、通知は特定されなければならない。コントロールの通知はその通知を送ったコントロールへのポインタを含む。このクラスはサブクラス無しで通常そのまま使用される。

コントロールのプレゼンター

コントロールのプレゼンターはコントロールを包むので、プレゼンテーション・データ・カプセルにより含まれることができる。それはすべてのプレゼンター・オブジェクトが提供する標準的な振る舞いを提供する。このクラスはサブクラス無しで通常そのまま使用される。

プレゼンテーションがオープンされたときとクローズされたときコールされるメソッド。それらはディフォルトでは何もしない。サブクラスはそれが包むオブジェクトのためのこれらのメソッドを処理しなければならない。コントロールに対して、これらのメソッドは直接委任されなければならない。プレゼンテーションがオープンされたとき、コントロールはその状態に影響を与えることができる通知に接続する。クローズされるとき、接続は切断される。

プレゼンテーションが活性化されるときと非活性化されるときコールされるメソッド。それらはディフォルトでは何もしない。サブクラスはそれが包むオブジェクトのメソッドを処理しなければならない。コントロールに対して、これらのメソッドは直接委任される。プレゼンテーションが活性化されるとき、いくつかのコントロールはアクティブのときだけ妥当となる通知に接続する。非活性化されときは接続は切断される。

TControlSelection（コントロール選択）

コントロール選択は、コントロール・プレゼンターの中に包まれプレゼンテーションの際に格納される単一のコントロールと付加的にその中のコマンドを特定する。

コントロール内のコマンドをアクセスするメソッド。これらはコマンドが特定されていなければ不当な値を戻す。

TUniControl（ユニコントロール）

ユニコントロールは、単一のコマンドを提供し、適切なユーザとの対話に続いて実行させられるコントロールのためのアブストラクト・ベース・クラスである。この形式のコントロールの例はボタンとチェックボックスである。

コントロールにより提供され実行されるコマンドを特定するメソッド。通知が

、コマンドが変更されたとき登録された接続に送られる。

コントロールがイネーブルとされるか否かに影響を与える通知者に接続するためとそれから切断するためにコントロールが使用するメソッド。ConnectEnabledNotifiersはコントロールがオープンされるときコマンドにより特定される通知者に接続する。DisconnectEnabledNotifiersは、コントロールがクローズされるときこれらの接続を切断する。

コントロールがEnable（イネーブル）とされるべきか否かに影響を与える何かが起きたことを示す通知を受信するメソッド。更新イネーブル（UpdateEnabled）はコマンドがアクティブか否かをチェックしてイネーブル（Enable）とディスネーブル（Disable）を適切にコールする。

コントロールのデータ値のプレゼンテーションに影響を与える通知者に接続するためとそれから切断するためにコントロールが使用するメソッド。ConnectDaTaNotifiersは、コントロールがオープンされるときコマンドにより特定される通知者に接続する。DisconnectDaTaNotifiersは、コントロールがクローズされるときこれらの接続を切断する。データ値を表示しないコントロール（例えばボタン）は、何もしないようにConnectDataNotifiersをオーバライドする。

TButton（ボタン）

ボタンは押されるとコマンドを実行するユニコントロールである。このクラスは通常サブクラス無しで使用され、単にコマンドをセットして離れる。

プレゼンテーションが活性化されるときと非活性化されるときにコールされるメソッド。プレゼンテーションが活性化されるとき、いくつかのコントロールはアクティブのときだけ適切な通知に接続する。非活

性化されるときには、これらの接続は切断される。プレゼンテーションが活性化されるとき、ボタンはキー等価通知のために登録する。この接続はプレゼンテーションが非活性化されるときに切断される。

コントロールのデータ値のプレゼンテーションに影響を与える通知者に接続しまたそれから切断してユーザを制御するメソッド。ConnectDataNotifiersはコン

トロールがオープンされるときコマンドにより特定される通知者に接続する。DisconnectDataNotifiersはコントロールがクローズされるときこれらの接続を切断する。データ値を表示しないコントロール（例えばボタン）は何もしないように接続データ通知ConnectDataNotifiersをオーバライドする。

チェックボックス

チェックボックスはブール値をセットするコマンドへのユーザ・インターフェイスである。適切なユーザとの対話に続いて、チェックボックスは、値を変更するようにコマンド・メソッドをコールしてそのコマンドを実行する。このクラスは通常サブクラス無しで使用され、単にコマンドとその値のゲッターとセッターをセットし、離れる。

スライダ

スライダは、単一浮動小数点値を表示して適切な

ユーザとの対話に続いて変更されることを可能とするユニコントロールである。スライダの例は第９図と第10図に示した。

TMultiControl（マルチコントロール）

マルチコントロールはいくつかのコマンドを提供するコントロールのためのアブストラクト・ベース・クラスであり、適切なユーザとの対話に続いてそれらを変更することを可能とする。この形式のコントロールの例はラジオ・ボタンとメニューである。

TRadioButton（ラジオ・ボタン）

ラジオ・ボタンは２あるいはそれ以上のブール値を表示するマルチコントロールであり、適切なユーザとの対話に続いてそれらを変更することが可能となる。ラジオ・ボタンは、第11図に示すように、正確に１つのボタンが選択されるという制約を有する。紙（Paper）が選択されると、1100でその円が黒くされる。プラスチック（Plastic）が選択されると、1110でその円が選択される。両方を選択されることはできない。

TCommand（コマンド）

コマンドは、オブジェクトへのあるいはオブジェクトのセットへのリクエスト

をカプセル化して特定のア

クションを行う。コマンドは通常、ボタンを押すことや、メニュー項目を選択すること、あるいは直接処理のようなエンド・ユーザ・アクションに応答して実行される。コマンドは、その出現を決定するためにコントロールにより使用することができるそれら自身についての種々の情報（例えば、名称グラフィック、キー等価、それらがアクティブか否か）を提供することができる。サブクラスは、コマンドがアクティブか否かを決定するために、現在の選択、アクティブ・ユーザ・インターフェイス要素、あるいは他のパラメータを調べるメソッドを処理しなければならない。サブクラスは、このコマンドがアクティブか否かに影響を与えることができる通知インタレストを戻すようにゲット・アクティブ・インタレストをオーバライドしなければならない。

第12図は本発明による詳細論理を描くフローチャートである。フローチャート論理は1200に進み、コントロールは機能ブロック1210にまで直接渡される。そこでコマンド・オブジェクトはメニューに加えられる。この機能ブロックにより実行されるステップは、1）コマンドからメニュー項目を生成するステップ、ここでメニュー項目はコマンドを含む他のオブジェクトデータ構造であるステップ、2）メニュー項目のリストにあるメニュー項目を追加するステップ、および3）メニューの出現が不当であることをデータ構造fValidに

マークするステップである。その後、メニューがプルダウンされると、出現はシステムの状態に基づいて再計算される。

各メニューはビューである。ビューはサイズと位置の情報を含む。各メニューはメニュー項目のリストを含む。各メニュー項目は現在の出現を反映するコマンドと変数を含む。変数は、メニュー項目がイネーブルであるか否か（Boolean fEnabled）、その名称（TTextLabel fName）、そのグラフィック（TGraphicLabel fGraphic）、およびその出現が現在妥当か否か（Boolean fValid）を含む。これらの変数の各々は、メニュー項目がいつ生成されたかをコマンドに訊ねることにより決定される。

次に、クエリーは、機能ブロック1220に示されるように、通知インタレストたためのコマンド・オブジェクトに送られる。各コマンドは異なる形式の通知に接続する４つのメソッドを有する。i）その名称に影響を与える通知、ii）グラフィックに影響を与える通知、iii）コマンドがアクティブか否に影響を与える通知、およびiv）いずれかのデータに影響を与える通知である。この場合、コマンドのために生成されたメニュー項目はアクティブな通知に接続する。それは接続オブジェクトをConnectActiveに渡すことにより行う。コマンドはその後、コマンドがアクティブか否に影響を与える通知者に接続オブジェクトを接続する。その

後、コントロールは機能ブロック1230に通され、メニュー項目を描くことが必要なときイネーブルな状態のコマンドをクエリーする。メニュー項目を描くために、メニュー項目はそのコマンドのためにメソッド“IsActive“をコールする。そのコマンドは、それが望むシステム状態がなんであるかを見て、決定ブロック1240に示されるように、現在のコンテキスト内でそれがアクティブか否かを戻す（例えば、いくつかのコマンドは、特定の形式のウインドウが正面にあるとき、あるいは特定の形式のオブジェクトが選択されるときにのみアクティブである）。その後、メニュー項目は機能ブロック1250および1260に示されるように、コマンドにより戻される値にマッチするように、その内部状態（各メニュー項目内のブール値）と出現を更新する。

ユーザ・アクションが入力ブロック1270に示されるようにコマンドを含むときは、ユーザは常にコマンドが処理されるようにする。これはメニュ項目、コントロール、あるいはオブジェクトの直接処理を介して行うことができる。このアクションは、機能ブロック1280に示すようにドキュメント状態が修正されるようにする。ドキュメントが通知を送るとき、以下のステップが実行される。1）ドキュメントにより送られた通知に接続されたいずれかのメニュー項目（あるいは他のコントロール）は、通知メッセージを受信する。

このメッセージはその通知に送られたオブジェクトへのポインタと共に変更の名称を含む。2）その後メニュー項目はその状態を更新する。コントロールは更な

る処理のため機能ブロック1230に戻される。

第13図は、本発明による表示を示す。メニュー項目は編集（Edit）1300であり、関連付けられた多数のサブメニュー項目を有する。アンドゥ（Undo）1310は、アクティブメニュー項目であり、関連する機能を実行するように選択することができる。リドゥ（Redo）1320は非アクティブであり、灰色で提供され、この時点では選択することができない。チェックボックスはまた、デバッグ用コントロール・パネル1350の一部として1360に示されている。

プレゼンテーション・テンプレートと持続性

データプレゼンテーションはテンプレートから生成され、ユーザ・インターフェイス・オブジェクト内の期間保存される。すべてのデータを有するものはシステム内ではモデルである。モデルはデータの処理を含み実行する。データ交換は、カット、コピー、ペースト動作を介して実行される。データの参照は、選択、アンカーおよびリンクにより提供される。データ・モデルは他の中に埋め込むことができる。ユーザは、関連するユーザ・インターフェイスにより提供されるプレゼンテーション（例えば、アイコン、寸描、フレー

ム、ウインドゥ、ダイアログ、コントロール・パネル）を介してモデルと対話する。データ・モデルはすべてのプレゼンテーションを生成し、他のオブジェクトへのアクセスメソッドを代理し、ユーザ・インターフェイスと呼ばれる。

ユーザ・インターフェイスは特定のモデルのためのプレゼンテーションの組、例えばアイコン、寸描、フレーム、ウインドゥを含むモデルである。要求されるとき、プレゼンテーションは望ましいプレゼンテーションの形式、ユーザ名、場所、他の基準とに基づいて既に生成されているものから選択される。望ましいプレゼンテーションが見つからないとき、新しいプレゼンテーションを生成して関連するアーカイブ（保管所）から１つをコピーすることにより、ユーザ・インターフェイスに加える。プレゼンテーションは持続的プレゼンテーション情報（例えば、ウインドゥ・サイズ場所、およびスクロール位置）がもはや要求されないときにはデリートされてもよい。

プレゼンテーションは、データを見て処理するために使用されるユーザ・イン

ターフェイス要素（例えば、メニュー、ウインドゥ、ツール）を包むプレゼンテーション可能なオブジェクトのセットを含む。プレゼンテーションはこれらのオブジェクトが提供するデータへの参照を提供する。プレゼンテーションは、プレゼンテーションが活性化されるときプレゼンテー

ション可能なオブジェクトをインストールしあるいは活性化する。同様に、これらのオブジェクトはプレゼンテーションが非活性化されるときに除かれ、あるいは非活性化される。プレゼンテーションはその目的に従って同定され、（例えばアイコン、寸描、フレーム、ウインドゥ）、後の選択のためまだ決定されるべき基準（例えばユーザの同一性）を保持する。

プレゼンテーションは、プレゼンテーションがオープンされるときあるいはアクティブであるときに画面上に表示され、あるいは他に使用可能なプレゼンテーション可能なオブジェクトの集まり（例えばユーザ・インターフェイス要素）から構成される。

プレゼンテーションは、アーカイブ（保管所）に含まれるテンプレート・プレゼンテーションから生成される。これらはユーザ・インターフェイス要素のようなオブジェクトから構成され、順にそれらはグラフィックとテキスト・ストリングのようなより小さいオブジェクトから構成される。

アーカイブはユーザ・インターフェイス要素（例えばウインドゥ、メニュー、コントロール、ツール）とプレゼンテーション（例えば、アイコン、寸描、フレーム、ウインドゥ）を含むテンプレート・オブジェクトを有するモデルである。

ダイアログ・ボックスとコントロール・パネル

コマンド・オブジェクトを異なる手法で用いることにより、コントロール群の２つの独立した振る舞いを制御することができる。第１は、それらが直ちにデータに影響を与えるか否か、あるいは設定が効果を上げる前に、ユーザがOKを押さなければならないか否かである。第２はそれらが互いに独立か否か、あるいは設定がアトミック（atomic）動作を提供するか否かである。

コントロールはコマンドを含む。ユーザがコントロールを処理するとき、コン

トロールはコマンド内にパラメータをセットしてそれが実行されるようにする。コマンドは選択により特定されるモデル・データに作用する。

イミーディエイト

データにすぐにに影響を与えるコントロールは、リアル・モデル・データを特定する選択を含むコマンドを有する。ユーザがコントロールを処理するときコマンドはこのデータを変更する。データが変わるとき、それは変更通知を送り、データの状態に依存するビューとコントロールは正確に現在の状態を反映する。

遅延

リアル・データを変更しないよう設計されたコントロールは代わりにプロトタイプのデータに作用しなければならない。リアル・モデル・データはユーザがOKボタンを押すような他のアクションを取るまで変更されない。これは２つの手法で達成される。

コントロールはコントロールそれ自身を特定する選択を含むコマンドを含む。ユーザがコントロールを処理するとき、コマンドはコントロールの値を変更するが、他のモデル・データを変更させない。ユーザがOKボタンを押すと、OKボタン内のコマンドは、ユーザが処理した各コントロール内の値に一致するようにリアル・モデル・データを変更する。

コントロールは、OKボタンにより含まれるコマンドのパラメータを特定する選択を含むコマンドを有する。ユーザがコントロールを処理するとき、コマンドはOKボタンのコマンドを変える。ユーザがOKボタンを押すとき、OKボタンのコマンドはそれ自身に含まれる値を一致させるようにリアル・モデル・データを変更する。

独立

互いに独立に動作するコントロールはコントロール・パネルあるいはダイアログ期間が完了した後、個々にアンドゥできるアクションを要求し表現する。

これはコントロールにより実行された場合の、コマンドの通常の振る舞いである。

アトミック（atomic）

コントロールの他のセットは一緒に働き、アトミック（atomic）動作としてアンドゥされ、リドゥされるように設計される。これは、ダイアログ・ボックスあるいはコントロールが開始されるときアンドゥ・スタック上にマークを置くことにより達成される。終了したとき、コントロール・パネルを解散させることによりあるいは（上記IIBのように）ユーザがOKボタンを押すときに、マークがアンドゥ・スタック上に置かれたときから実行されるコマンドのすべてが単一のコマンド群に集められる。この群は単一群としてアンドゥあるいはリドゥされることができる。

キャンセル

（上記IIBのように、通常OKボタンに伴う）CANCEL（キャンセル）ボタンを含むコントロール・パネルは、上記IIIBで述べた技術と同様な技術である。マークは、ダイアログ・ボックスあるいはコントロール・パネルが開始されるときアンドゥ・スタック上に置かれる。ユーザがCANCELボタンを押すと、マークがアンドゥ（Undo）・スタックに置かれる前のすべてのコマンドがアンドゥ（Undo）される。この技術はコントロール

がデータに直ちに影響を与えるか否かにかかわらず働く。

ダイアログ・ボックス内での

アトミック（atomic）コマンドの実行

他のオブジェクトあるいはデータを処理するようユーザが対話するオブジェクトはコントロールと呼ぶ。コントロールの例は、メニュー、ボタン、チェックボックス、およびラジオ・ボタンである。各コントロールはコマンドを含み、それはエンド・ユーザ・アクションを実行する。コマンドは選択オブジェクトにより特定されるデータに作用する。ユーザがコントロールを処理するとき、それはコマンド内にパラメータをセットしてそれが実行されるようにする。このようにして、データ値を変更する。

互いに独立に働くコントロールは、コントロール・パネルあるいはダイアログ期間が完了したあと、個々にアンドゥされることができる表現アクションを要求

する。これは、コマンドがコントロールにより一旦実行されたとき通常の振る舞いである。他の組のコントロールは共に動作するように設計され、アトミック（atomic）動作に従ってアンドゥされリドゥされるべきである。これはこの特許の主題である。

アトミック（atomic）実行の詳細な論理は第14図に提供されるフローチャートに示される。処理は端子1400

に進み、コントロールは機能ブロック1410にすぐに通され、ダイアログ・ボックスが活性化される。ダイアログ・ボックスが活性化されるとき、マークはアンドゥ・スタック上に置かれる。アンドゥ・スタックはユーザが実行したすべてのコマンドのリストである。アンドゥが押されると、スタックの最上部のコマンドがアンドゥされる。すぐにリドゥされなければ、捨て去られる。その後、機能ブロック1410で、コントロールのユーザ処理が検出される。コントロールの処理が機能ブロック1430で述べたように適切にコマンドのデータ値を変更して、コントロールを実行する。例えば、チェックボックスは０と１の間でコマンドのｆチェック（fChecked）フィールドを行き来する。最後に、コマンドはアンドゥ・スタック上に記録され、機能ブロック1440に示されるように、続いてアンドゥされることができる。

判定ブロック1450で検出されるように、ユーザが続いてダイアログ・ボックス内の各コントロールを処理すると、コントロールは機能ブロック1430に進む。しかしながら、判定ブロック1460で検出されるように、ユーザがOKボタンを押したときは、コントロールは機能ブロック1420に進む。最後に、ダイアログ・ボックス内の各コントロールがユーザの満足に達したとき、ユーザはOKボタンを押す。マークが機能ブロック1440でアンドゥ・スタック上に置かれて、以来実行された

すべてのコマンドは単一のコマンド群内に集められ、機能ブロック1470で示されるようにアンドゥ・スタック上に戻される。コマンド群は多くのコマンドを一緒に集めるコマンドである。実行され、アンドゥされ、あるいはリドゥされるとき、コマンド群は順番に各コマンドを実行し、アンドゥし、あるいはリドゥする。

コマンド群は、その後、単一アトミック（atomic）動作としてアンドゥされあるいはリドゥされることができるアンドゥ・スタック上に戻される。

ダイアログ・ボックス内の遅延された

コマンド実行

ユーザが他のオブジェクトあるいはデータを処理するように対話するオブジェクトは、コントロールと呼ばれる。コントロールの例はメニュー、ボタン、チェックボックス、ラジオ・ボタンである。各コントロールはコマンドを含み、それはエンドーユーザのアクションを処理する。コマンドは選択オブジェクトにより特定されるデータに作用する。ユーザがコントロールを処理するときそれはコマンド内にパラメータをセットしてそれが実行されるようにする。これにより、データ値が変更される。ユーザが他のアクションを実行するまでデータが遅延して変更されることは、本発明の１つの特徴である。例えば、ダイアログ・ボックス内のコントロールはユーザがOKボタンを押す

まで、データ値を変更しようとしない。

コントロールが生成されるときコマンドが特定されなければならない。コントロールはこのコマンドのコピーを作りそれをフィールドfCommandに格納する。コマンドが何らかのデータ値を供給するならば、コマンドの適切なゲット（Get）メソッドとセット（Set）メソッドへのポインタがまた特定されなければならない。コントロールはこれらのメソッド・ポインタをfGetMethodとfSetMethodのフィールドにそれぞれ格納する。コマンドにより修正されるデータは選択オブジェクトにより特定される。通常、この選択オブジェクトはリアル・モデル・データを特定する。代わりに、OKボタンのコマンド内のデータ値を特定する選択オブジェクトであってもよい。

ユーザがコントロールを処理するとき、コントロールのコマンドが実行されOKボタンのコマンド内のデータ値が変更される。ユーザがダイアログ・ボックス内の各コントロールを処理するとき、コントロールコマンドが実行されて、OKボタンのコマンド内のデータ値が変更される。このようにして、ユーザがOKボタンを押すと、OKボタン内のコマンドがコントロールのコマンドにより処理されるよう

にそれ自身内に含まれるデータ値に一致するようにリアル・モデル・データを更新する。この処理はコントロールの処理が完了するまで繰り返される。

ラベル

ラベルはグラフィックあるいはテキスト・ストリングを含むグラフィック・オブジェクトである。それらはウインドゥ、メニュー、ボタン、および他のコントロールを同定するために使用される。ラベルはそれらのコンテナの状態に従ってそれらの外見を変えることができる。それらは中間明度の灰色のバックグラウンド上に描かれ、特別の状態を表示する必要のないときに自然に現われる。ラベルは、非アクティブのとき、禁止されているとき、あるいは選択されたとき、その外見を変える。

非アクティブ

ウインドゥ・タイトルはウインドゥが正面最上部でないとき非アクティブにセットされる。同様に、コントロールのラベルはコントロールが正面最上部のウインドゥあるいはコンテナ内にないとき非アクティブにセットされる。グラフィック・ラベルは、薄暗く出現させるために、非アクティブのとき55％の白と混ぜられている。テキスト・ラベルでは、非アクティブペイントはHSVカラー・モデルの飽和成分を処理することにより自然のペイントから導かれる。飽和は、非アクティブのとき0.45倍される。

禁止（ディスエーブル）

コントロール・ラベルはコントロールが特定のコンテキスト中で適用しないときには薄暗くされる。グラフィック・ラベルは非アクティブのときには薄暗くするため46％の白と混ぜられている。テキスト・ラベルでは、禁止ペイントはHSVカラー・モデルの飽和成分を処理することにより自然のペイントから導かれる。飽和は、禁止のときに0.54倍される。

選択

コントロール・ラベルはコントロールが処理されるにつれてハイライトにされる。グラフィックとテキストはそれらの自然状態で描かれ、ハイライト状態のと

きに白いバックグラウンド上に描かれる。

スマート・コントロール・ラベル

（Smart control Label）

コントロールはオブジェクトあるいはデータの現在の状態を決定するためのコマンドを使用する。ユーザとの適切な対話に続いて、コントロールはコマンド・パラメータを更新してそれが実行されるようにする。例えばチェックボックスはコマンド・パラメータをオンあるいはオフにセットして、そのコマンドを実行してデータ値を変更する。コントロールはその機能を示すようにラベルを表示する。このラベルはグラフィッ

クあるいはテキスト・ストリングを含むグラフィック・オブジェクトである。コントロールが状態を変えるとき、ラベルは、開発者が付加的なコードを書くことを要求することなく自動的にその出現を変える。これらの状態は、アクティブ／非アクティブ、イネーブル／ディスエーブルおよび選択／非選択を含む。

第15図はスマート・ラベル処理と関連する詳細論理を示し、その処理は開始ターミナル1500に進む。そこでは、コントロールはスマート・ラベルの初期化のためすぐに1510に進む。コントロールが生成されるとき、そのラベルはその関連するコマンドにより提供されるテキスト・ストリングあるいはグラフィックで初期化される。各コマンドはこの目的のためGetGraphicとGetNameと呼ばれるメソッドを提供する。コントロールはメソッドSetActiveをコールすることにより現在アクティブか非アクティブかをラベルに知らせる。同様に、コントロールはSetEnabledメソッドをコールしてそれがイネーブルであるか否かをラベルに知らせる。また、SetSelectedメソッドをコールしてユーザにより現在選択されているか否かをラベルに知らせる。

スマート・ラベル処理での次のステップはラベルが描かれるとき機能ブロック1520で起きる。コントロールが活性化されているとき、それはそのラベルの描写（Draw）メソッドをコールしてラベルをその画面上に出

現させる。非アクティブならばラベルは通常よりも薄暗く描かれる。これはHSV

カラー・モデルの飽和成分を処理することによりなされる。飽和は非アクティブのとき0.45倍される。ディスエーブルならば、ラベルは通常よりも薄暗く描かれる。これはHSVカラー・モデルの飽和成分を処理することによりなされる。飽和はラベルがディスエーブルのとき0.54倍される。選択されたならば、ラベルはハイライトされたバックグラウンド上に出現させられる。ラベルは通常、中間色グレーのバックグラウンド上に描かれる。ハイライトされるときラベルは白いバックグラウンド上に描かれる。他には、ラベルは通常通り描かれる。

次の処理はラベルが機能ブロック1530に示されるように活性化されている／非活性化されているとき起きる。コントロールが活性化されているあるいは非活性化されているとき、それはSetActiveメソッドをコールしてラベルに知らせる。コントロールは、再描画される必要がある画面の一部を示すアーギュメントつきで無効化（Invalidate）をコールすることによりその出現を更新することが必要であることを示す。その後、機能ブロック1540で、処理はコントロールがイネーブルあるいはディスエーブルされるとき起きる。コントロールは、イネーブルあるいはディスエーブルされるとき、SetEnabledメソッドをコールすることによりラベルに知らせる。その後、コントロールは、再描画さ

れる必要がある画面の一部を示すアーギュメントつきで無効化（Invalidate）をコールすることによりその出現を更新することが必要であると示す。

テストが判定ブロック1550で実行され、コントロールが選択されているか非選択であるかを判定する。コントロールが選択されあるいは非選択であるときは、SetSelectedメッソド　をコールすることによりラベルに知らせる。その後、コントロールは、再描画される必要がある画面の一部を示すアーギュメントつきでInvaliate（無効化）をコールすることによりその出現を更新することが必要であると示し、コントロールは更なる処理のため機能ブロック1520に進む。

スマート・ウインドゥ・ラベル

（Smart Window Label）

タイトルはその目的を示すためにウインドゥ内に表示される。例えばドキュメントを編集するためのウインドゥのためのタイトルは通常ドキュメントの名称で

ある。ラベル・オブジェクトはタイトルの追尾を保つために使用される。このラベルはグラフィックあるいはテキスト・ストリングを含むグラフィック・オブジェクトである。ウインドゥが状態を変えると、ラベルは、開発者が付加的なコードを書くことを要求することなく、その出現を自動的に調整する。ウインドゥはアクティブあるいは非アクティブのどちらかであ

る。スマート・ウインドゥ・ラベル処理は第16図のフローチャートで示され、その詳細論理はそれを参照して説明される。

第16図の処理はターミナル1600に進み、そこでコントロールはタイトルが初期化されるべきために機能ブロック1610にすぐに進む。ウインドゥ・タイトルはウインドゥが生成されるとき開発者により特定される。このタイトルはfTitleと呼ばれるTLabelオブジェクト内に格納される。コントロールは、メソッドSetActiveをコールしてアクティブかあるいは非アクティブかをタイトルに知らせる。その後機能ブロック1620に進む。ウインドゥが描かれるとき、そのfTitleオブジェクトのDrawメソッドをコールしてタイトルが画面上に出現するようにする。非アクティブのときタイトルは通常より薄暗く描かれる。これはHSVカラー・モデルの飽和成分を処理することによりなされる。飽和はラベルが非アクティブのとき0.45倍される。他の場合には、タイトルは通常通り描かれる。

次のステップはタイトルが機能ブロック1630で活性化されている／非活性化されているとき処理される。ウインドゥが活性化されているあるいは非活性化されているとき、それはSetActiveメソッドをコールしてそのfTitleオブジェクトに知らせる。その後、ウインドゥは、再描画される必要がある画面の一部を示すアーギュメントつきでInvalidateをコールすることに

よりその出現は更新することが必要であると示す。その後、その新しい状態を反映するようにタイトルを再描画するためにコントロールは機能ブロック1620に戻る。

デコレーション

ユーザ・インターフェイス要素の多くの視覚的測面は、多くの要素間で共通で

ある。例としてシャドウ、境界、およびラベルがある。個々の可視的特徴はデコレーションとして参照される。デコレーションは他のグラフィックスと結合されてウインドゥとコントロールのような特定のユーザ・インターフェイス要素の可視的形状を形成する。本発明の主題は多くのことなる形式のデコレーションをサポートすることである。

バックグラウンド

他のオブジェクトの後ろに描かれるデコレーションはバックグラウンドと呼ばれる。１つの形式のバックグランドは周囲の描画面とフラッシュするように描かれる。それはフレームつきであるいはフレーム無しで描かれてもよい。他の形式のバックグランドはハイライトとシャドウつきで描かれ、それは回りの描画面から持ち上げられているように見える。最後の形式のバックグランドはハイライトとシャドウつきで描かれ、それは周囲の描画面より後退して見える。

これらのバックグランドの使用例はボタンである。通常ボタンを述べるテキストあるいはグラフィックは持ち上がったバックグランドの上に描かれる。ユーザにより押されれると、テキストあるいはグラフィックは後退したバックグランド上に再描画される。他のウインドゥがアクティブのときのようにボタンが非アクティブならば、ボタンのテキストあるいはグラフィックはフラッシュ・バックグランド上に薄暗く描かれる。

境界

他のオブジェクトあるいはエリアを囲むデコレーションは境界と呼ぶ。境界の例はフレームとシャドウである。フレームはリアル・ワールド内でのペイントを包み込むフレームのように他のグラフィックを囲む境界である。バックグランドのように、フレームは、周囲の描画面よりしたに後退したように、フラッシュするように、あるいはそれより持ち上がったように描くことができる。シャドウはオブジェクトの回りにシャドウを加える特定の形式の境界であり、周囲面より浮いているかのようにそれを見せる。

ユーザ・インターフェイス要素の多くの視覚的測面は、多くの要素間で共通である。例としてシャドウ、境界、およびラベルがある。これらの個々の視覚的特

徴の各々はデコレーションとして参照される。デコ

レーションは他のグラフィックスと結合されてウインドゥとコントロールのような特定のユーザ・インターフェイス要素の視覚的形状を形成する。あるデコレーションはハイライトとシャドウを用いて周囲描画面より上にあるいは下にあるかのように見せる。デコレーションはこれらのハイライトとシャドウのペイントを自動的に導くことができる。

塗りつぶし（Fill Paint）

塗りつぶしはデコレーションの基本的カラーを表す。すべての他のペイントは塗りつぶしから導かれる。塗りつぶしはfFillPaintと呼ばれるカラー・フィールド内に管理者により格納される。塗りつぶしは通常、デコレーションが生成されるときに作成者により特定される。しかしながら、カラーが特定されないと、中間色グレーが選択される。

フレーム・ペイント（Frame Paint）

フレーム・ペイントはデコレーションの回りに線を描くために使用され可視的なコントラストを提供する。フレーム・ペイントはfFramePaintと呼ばれるTColorのフィールド内にデコレーションにより格納される。フレーム・ペイントはデコレーションが生成されるとき開発者により特定されてもよい。しかしながら、フレーム・ペイントが特定されないと、塗りつぶ

しから自動的に計算される。これはHSVカラー・モデルの飽和と値成分を処理することによりなされる。飽和は最大値が１として、４倍される。値は４で割られる。

ハイライト・ペイント（Highlight Paint）

ハイライト・ペイントは、オブジェクトが実際の３次元オブジェクトであるならば光がそのオブジェクトにあたる線を描くために使用される。ハイライト・ペイントはfHighlightPaintと呼ばれるTColorフィールド内にデコレーションにより格納される。ハイライト・ペイントはデコレーションが生成されるとき開発者により特定されてもよい。しかしながら、ハイライト・ペイントが特定されない

と、塗りつぶしから自動的に計算される。これはHSVカラー・モデルの飽和と値成分を処理することによりなされる。飽和は0.8倍される。値は最大値を１として、1.25倍される。

シャドウ・ペイント（Shadow Paint）

シャドウ・ペイントは、オブジェクトが実際の３次元オブジェクトであるならば光がそのオブジェクトにあたる線を描くために使用される。ハイライト・ペイントはfHighlightPaintと呼ばれるTColorフィールド内にデコレーションにより格納される。ハイライト・ペイントはデコレーションを生成するとき開発者が特

定してもよい。しかしながら、ハイライト・ペイントが特定されないと、塗りつぶしから自動的に計算される。これはHSVカラー・モデルの飽和と値成分を処理することによりなされる。飽和は0.8倍される。値は最大値を１として、1.25倍される。

セマンティックからの入力シンタクスの分離

グラフィック・ユーザ・インターフェイスはマウスを動かし、それらを選択するためにオブジェクトをクリックし、移動またはコピーのためにオブジェクトをドラッグし、その後それらをオープンするためにダブル・クリックすることにより処理される。これらの操作は直接操作あるいは対話と呼ばれる。マウスをユーザが押し、動かし、離すことに対応するイベントのシーケンスは入力シンタクスと呼ばれる。あるイベント・シーケンスは、セマンティック操作と呼ばれる特定のアクションを示すために使用される。

セマンティック操作を処理するコードから入力シンタクスを理解するためのコードを分離することは本発明の主題である。この処理は相互作用（対話，Interacts）と相互作用可能（対話可能，Interactable）と呼ばれるオブジェクト内にそれぞれ埋め込まれる。第17図はオブジェクトがどのようにして生成され、動かされ選択されることができるオブジェクトとの一般的な対話の間でどのように互いに通

信するかを示す。

処理はターミナル1700に進み、そこでコントロールは機能ブロック1710にすぐに通され、マウス・ボタンが押されたかどうかを決定する。マウス・ボタンが押された位置で画面の一部に対して応答可能なオブジェクトにイベントが送られる。このオブジェクトはビュー（View）と呼ばれる。その後、機能ブロック1720で、インタラクタ（Interactor）が入力シンタクスを説明するために生成される。これはビューのメソッド・インタラクタ生成（CreateInteractor）をコールすることによりなされる。インタラクタ（Interactor）が生成されると、可能なユーザ・アクションを実行するオブジェクトへのポインタがパラメータとして渡される。

この議論のために、ユーザが、選択され動かされることができるオブジェクト上でマウス・ボタンを押したとする。この場合、選択を実行するオブジェクトと目標オブジェクトに向かって移動するオブジェクトとがパラメータとしてインタラクタ（Interactor）に渡される。初期のビュー（View）はこれらの振る舞いの両方を実行することができるであろう、あるいはそれらは１つあるいは２つの別のオブジェクトにより実行されることができるであろう。オブジェクトあるいは複数のオブジェクトは相互動作可能（Interactable）と統括的に呼ばれる。

インタラクタ（Interactor）は機能ブロック1730で開始される。この処理はインタラクタ（Interactor）Viewに戻してインタラクタ（Interactor）の処理を進める。これはInteractorStartメソッドをコールしてパラメータとして初期マウス・イベントを渡すことによりなされる。StartメソッドはfInitialEventに初よマウス・イベントを保存する。インタラクタ（Interactor）は変数fIntercation Typeを定数kSelectにセットすることにより選択（Select）モードにはいる。それはそのSelectBeginメソッドをコールすることにより選択動作を開始するようインタラクタブル（Interactable）に依頼する。

その後、インタラクタ（Interactor）は、短い時間待って、機能ブロック1740で示されるように、通過する。新しいマウスイベントが、マウスの現在の状態を示す時間となったとき、インタラクタ（Interactor）に送られる。その後、システムはマウスが判定ブロック1750でまだ押されていることを検出すると、コント

ロールは機能ブロック1740にまで通される。他に、コントロールはターミナル1760に通される。マウス・ボタンがまだ押されていれば、インタラクタ（Interactor）はまだ正しい状態にあると確信して、Interactableに正しい操作を実行するよう依頼する。インタラクタ（Interactor）はfInteractionがkSelectionであれば、Selecting（選択中）である。

fInteractionTypeがkMovingであれば、それはMoving（移動中）である。

選択（select）中ならば、インタラクタ（Interactor）は現在のマウス位置と初期マウス位置を比較する。現在のマウス位置はGetCurrentLocationをコールすることにより得られる。初期マウス位置はGetInitialLocationをコールすることにより得られる。２つが同じかあるいは多少しか異ならなければ、ユーザはまだオブジェクトを選択している。その後、インタラクタ（Interactor）は、そのSelectRepeatメソッドをコールすることにより選択操作を継続するようInteractableに依頼する。しかしながら、２つの点が予め決められたスレッシュホールド以上に異なるときには、ユーザはオブジェクトを動かし始めている。この場合、インタラクタ（Interactor）はそのSelectEndメソッドをコールすることにより選択操作を終了するようInteractableに依頼する。その後、その移動開始メソッドをコールして移動動作を始めるようにInteractableに依頼する。各々の場合、現在のマウス位置は引数（argument）として渡される。Moving（移動中）ならば、インタラクタ（Interactor）はMoveRepeatをコールして移動動作を継続するようインタラクタブルInteractableに依頼する。それは引数（argument）として現在のマウス位置を渡す。

ユーザがマウス・ボタンを離すとき、それは現在の

動作の終了を知らせる。Selecting（選択中）ならば、インタラクタ（Interactor）は、そのSelectEndメソッドをコールして選択動作を終了するようInteractableに依頼する。移動中ならば、インタラクタ（Interactor）はそのMoveEndメソッドをコールして移動動作を終了するようInteractableに依頼する。

局所化プレゼンテーション

局所化はアプリケーションを更新するプロセスであり、特定の場所のユニークな表現に従う。それは、言語の翻訳、グラフィック置換、およびインターフェイス要素の再配向を含む。例えば、ラベル、タイトル、およびメッセージの中で使用されるテキストは選択された言語に依存する。その方向と配向はメニュー、メニュー・バー・タイトル、スクロールバーあるいはツールバーの配置と配向に影響を与える。同様に、アイコンと他のグラフィックシンボルの選択は開発に依存する。不幸にも、メモリ内にユーザ・インターフェイス要素の多くの局在化されたバージョンをもつことは非常に高いものとなる。代わりに、ユーザ・インターフェイス要素の局在化バージョンは、メモリ内で要求されるまでディスク上に保たれる。

さらに、ユーザ・インターフェイス要素のすべての追尾を保ち、どのバージョンが使用されるべきかを決定することは非常にエラーとなりやすく高価である。

代わりに、ユーザ・インターフェイス要素が要求されるとき適切なものが現在の言語と他の文化的パラメータに従ってシステムにより自動的に選択しメモリに読み込む。

一旦局所化されると、ユーザ・インターフェイス要素はディスク・ディクショナリ内に格納される。ディスク・ディクショナリは、キーに与えられるとき、それをディスク内にあるいはディスクから読んだあとに値を戻るオブジェクトである。このディスク・ディクショナリは保管所（アーカイブ）と呼ばれるオブジェクトにより管理される。アーカイブは特定の表現を構成する個々のユーザ・インターフェイス要素を一緒に置く。適当なユーザ・インターフェイス要素の選択のプロセスは第19図に提供される。

処理はターミナル1900に進み、ユーザがプレゼンテーションをリクエストするときすぐに機能ブロック1910に通される。TOpenPresentationCommandはデータ・モデルに送られ、ユーザがこのデータをビューあるいは編集したいということを示す。このコマンドはTOpenPresentationCommand呼ばれる。プレゼンテーションはユーザがあるデータをビューしあるいは編集することを可能とするユーザ・インターフェイス要素の組である。プレゼンテーションはユーザインターフェース

オブジェクト内の期間に渡って格納され、このようにしてユーザのために連続性を維持する。ユー

ザ・インターフェイス要素はメモリ内で必要となるまでディスクに格納される。それらはユーザがリクエストしたデータプレゼンテーションの一部として要求され、あるいは翻訳あるいは他の局所化プロセスのために必要とされることがある。各ユーザ・インターフェイス要素はその要素をユニークに参照するIDを含む。しかしながら、同じユーザ・インターフェイス要素のすべての局在化バージョンは単一のIDを共有する。

局在化バージョンを異ならせるために、特定の言語、記述方向、および他の文化的パラメータが各局在化されたユーザ・インターフェイス要素と共に格納される。一緒に、これらのパラメータは場所として参照される。ユーザ・インターフェイス要素のすべてはファイルに格納される。このファイルは１つあるいはそれ以上のキー／値の対付きで、ディクショナリと同様に組織化される。そのキーはIDと場所を結合するオブジェクトである。値はユーザ・インターフェイス要素それ自身である。

新しいプレゼンテーションが機能ブロック1920で次に生成されなければならない。適切なプレゼンテーションは既に存在しないので、新しいものがユーザ・インターフェイス、アーカイブによりテンプレートから生成されなければならない。新しいプレゼンテーションはそのCreatePresentationメソッドをコールす

ることによりアーカイブの中に格納されるテンプレートから生成される。プレゼンテーションの形式はパラメータとしてこのメソッドに渡される。この形式は、表示されるべきデータの形式、それがそれ自身のウインドゥあるいは他のプレゼンテーションの一部の中に存在すべきか否か等の情報を含む。最後に、機能ブロック1930で、アーカイブは場所に従ってユーザ・インターフェイス要素を選択して、プレゼンテーションを構築する。アーカイブが特定の形式のプレゼンテーションを構築することができれば、それはプレゼンテーションを構築しユーザ・インターフェイスオブジェクトにこれを戻す各ユーザ・インターフェイス要素を一

緒に集める。

アーカイブが構築することができる各プレゼンテーションでは、それはプレゼンテーションを一緒に構築するユーザ・インターフェイス要素のリストをもつ。ユーザ・インターフェイス要素はコールされたディスク・ディクショナリにより維持されているディスク上に格納される。キーを与えると、ディスク・ディクショナリは対応するユーザ・インターフェイス要素を戻す。ユーザ・インターフェイス要素のIDはこのキーの基本的部分を構成する。キーの第２の要素は望ましい場所である。場所は、ユーザの自然言語と他の文化的属性を特定するオブジェクトである。場所は、参照サーバ（Preferences Server）からアーカイブにより自

動的に得られる。このサーバーはユーザに関連する個々の好みのすべてを含む。

好みのサーバーから得られる場所はIDと共に、TUserInterfaceElementKeyと呼ばれる単一のオブジェクトの中に結合される。このキーはパラメータとしてディスク・ディクショナリのGetValueメソッドに渡される。一致するIDと場所もつユーザ・インターフェイス要素が見つけられると、プレゼンテーションの一部として戻され含まれる。他に、場所パラメータはキーから省略されなければならず、あるいは他の場所がユーザ・インターフェイス要素のが見つけられるまで特定されなければならない。

対話フレームワーク・システム

オブジェクト指向オペレーティング・システムのグラフィック・ユーザ・インターフェイスのユーザは、しばしば、マウスを動かし、オブジェクトを選択するためにクリックし、移動あるいはコピーのためにオブジェクトをドラッグし、その後オブジェクトをオープンするためにダブル・クリックする。これらの動作は直接操作あるいは対話と呼ばれる。マウスをユーザが押し、移動し、離すことに対応するイベントのシーケンスは入力シンタクスと呼ぶ。あるイベント・シーケンスが特定のアクションを示すために使用され、セマンティック動作と呼ばれる。本発明は、Select（選

択），Peek（ピーク），Move（移動），AutoScroll（自動スクロール）およびDr

ag/Drop(Copy)（ドラッグ／ドロップ）（コピー）をサポートするオブジェクトのセマンティック動作に入力シンタクスを翻訳するための方法と装置を開示する。

本発明は、マウス・ボタンの押下を検出し以下の論理を採用する。

(a) ユーザがマウス・ボタンを押したときオプションキーが押されていたら、システムは変数fInteractionTypeを定数kDragにセットすることによりドラッグ・モードに入る。その後システムは動作の目標として選択されたオブジェクトを用いてドラッグ動作を進める。あるいは、

(b) オプションキーが押されていなかったならば、システムは変数fInteractionTypeを定数kSelectにセットすることにより選択モードに入る。その後、選択動作が進められる。

ユーザが既にマウス・ボタンを押していて押したままに保っているときには、以下の論理が関係する。システムは選択モードにあり、システムは最初にユーザがマウスを、移動スレッシュホールドと呼ばれるあるスレッシュホールド以上に動かしたか否かを判定する。これは、GetInitialLocationメソッドにより戻される初期のマウス位置をGetCurrentLocationメソッドにより戻される現在のマウス位置と比較することによ

りなされる。マウスが移動スレシュホールド以上に動かされていればシステムは選択モードを終了し移動モードに入る。それは変数InteractionTypeを定数kMoveセットすることにより行う。システムはその後そのSelectEndメソッドをコールして選択動作を終了するようオブジェクトをクエリーする。その後、システムはそのMoveBeginメソッド　をコールすることにより移動動作を開始する。

他に、マウスが動いていないときには、システムはマウスがどのくらい長く押されたままかをチェクするGetCurrentTimeメソッドにより戻される現在の時間とを比較することによりなされる。マウスがピーク（peek）スレシュホールドと呼ばれるあるスレシュホールド以上押下されたままならば、システムは選択モードを終了しピーク・モードに入る。それは変数fInteractionTypeを定数kPeekにセットすることにより行う。それは、そのSelectEndメソッドをコールして選択動

作を終了するようオブジェクトに依頼して、そのPeekBeginメソッドをコールしてピーク動作を開始する。他に、マウスが動かされないとき、あるいはあるピーク・スレシュホールドを越えて押されなときは、システムはオブジェクトのSelectRepeatをコールして選択動作を継続する。システムがユーザが移動モードにあることを検出したときは、システムはまずユーザがウインドゥ内で、あるいはウインドゥの境界

上で、あるいはウインドゥの外でマウスを動かしたか否かを判定する。それは、GetCurrentLocationメソッドにより戻される現在マウス位置とGetContainerBoundsにより戻されるオブジェクトのコンテナの境界とを比較して行う。

マウスがまだウインドゥの境界内にあれば、システムはオブジェクトのMoveRepeatメソッドをコールして移動動作を継続する。マウスがウインドゥの境界上にあれば、これは自動スクロール動作を示す。システムはマウス位置により示される方向にスクロールするようにオブジェクトのコンテナに依頼する。これは、コンテナのAutoScrollメソッドをコールして、現在のマウス位置をパラメータとして通すことによりなされる。一旦完了すると、システムは、オブジェクトのMoveRepeatメソッドをコールして移動動作を継続する。

マウスがウインドゥの外にあれば、システムは移動動作を終了してドラッグ・モードに入る。それは、変数fInteractionTypeに定数kMoveをセットすることにより行う。それは、そのMoveEndメソッドをコールして移動動作を終了するようオブジェクトに依頼する。それは、そのDragBeginメソッドをコールしてドラッグ動作を開始するようにオブジェクトに依頼する。システムがドラッグ・モードにあるときには、システムはオブジェクトのDragRepeatメソッドをコールしてド

ラッグ動作を継続する。システムがピーク・モードにあるときには、システムは最初に、移動スレッシュホールドと呼ばれるあるスレッシュホールドを越えてマウスをユーザが移動したか否かを判定する。これは、GetInitialLocationメソッドにより戻される初期マウス位置とGetCurrentLocationメソッドにより戻される現在マウス位置とを比較してこれを行う。

マウスが移動スレシュホールドを越えて動いていたときには、システムはピーク動作を終了して移動モードに入る。それは、変数fInteractionTypeを定数kMoveにセットすることにより行う。それは、そのPeekEndメソッドをコールしてピーク動作を終了するようオブジェクトに依頼する。それは、そのMoveBeginメソッドをコールして移動動作を開始する様にオブジェクトに依頼する。他に、マウスが動かされていないときには、システムはオブジェクトのPeekRepeatメソッドをコールしてピーク動作を継続する。

システムがマウス・ボタンをユーザが離したことを検出したときには、選択モードにあれば、システムは選択モードを終了する。それは変数fInteractionTypeを定数kNoneにセットすることにより行う。システムはそのメソッド選択終了をコールして選択動作を終了するようオブジェクトにクエリーする。システムが移動モードにあるときには、システムは移動モードを終

了する。それは変数fInteractonTypeを定数kNoneにセットすることによりこれを行う。その後、システムはそのMoveEndメソッドをコールして移動動作を終了するようオブジェクトにクエリーして、変数fInteractonTypeを定数kNoneにセットすることによりドラッグ・モードを終了する。システムがピーク・モードにあるときには、システムはピーク・モードを終了する。それは変数fInteractonTypeを定数kNoneにセットすることにより行う。それは、そのPeekEndメソッドをコールしてピーク動作を終了するようオブジェクトに依頼する。

ユーザがスクロールバーを動かすにつれて動的にウインドウの内容を更新することを可能とするハードウェアとソフトウェアのシステムを提供することが本発明の基本的目的である。システムはユーザがスクロールバーを押したことを検出する。ユーザがスクロールバーを押したとき、システムはスクロール・コマンドの初期化を始め、ウインドウ内に表示されているデータ部分を変更する。コマンドはスクロールのようなエンド・ユーザ・アクションを処理するオブジェクトである。スクロール・コマンドは１つのパラメータ、内容のビューがスクロールされるべき位置を有する。システムはこの位置を現在のスクロール位置にセットする。これは、コマンドのSetScrollPositionメソッドをコールして、スクロール

バーのメソッドの

GetScrollPositionにより戻される値に位置にスクロールを設定することによりなされる。

ユーザがスクローバー内でマウスを動かすときには、システムはスクロール・コマンドの実行を継続してウインドゥ内に表示されているデータ部分を動的に変更する。システムはコマンドのスクロール位置を新しいスクロール位置にセットする。この処理はコマンドSetScrollPositionをコールしてスクロールバーのメソッドのGetScrollPositionにより戻される値に等しく値をセットすることによりなされる。コマンドの実行は、その後、そのDoRepeatメソッドをコールすることにより繰り返される。これによりビューの内容が新しい位置にスクロールされる。この処理はユーザがマウス・ボタンを押し続けている間継続する。

ユーザがマウス・ボタンを離したときには、システムはスクロール・コマンドの実行を終了してウインドゥ内に表示されているデータ部分を動的に変更する。システムはコマンドのスクロール位置を最終スクロール位置にセットする。この処理はコマンドSetScrollPositionをコールしてスクロールバー・メソッドのGetScrollPositionにより戻される値に位置にそれが等しくなるようにセットすることによりなされる。

第20図は本発明によるスクロールと関連する詳細論理を示すフローチャートである。処理はターミナルブ

ロック2000に進み、すぐに機能ブロック2010に通される。そこで、現在のスクロール位置が現在のカーソル位置に基づいて初期化される。その後、判定ブロック2020で、スクロールバーつまみ（scrollbar thumb）選択されたかどうかを検出するようにテストを行う。スクロールバーつまみの例を第21A図にラベル2110に示す。スクロールバーつまみが選択されたときには、コントロールは、スクロールバーが動かされたか否かを判定するために判定ブロック2030に通される。そうならば、スクロール位置はスクロールバーつまみの新しい位置に送られ、表示がすぐスクロール動作に反映して再フォーマットされユーザのために表示がなされ

る。スクロールバーつまみが動いていなければ、他のテストが判定ブロック2050で行われ、スクロールバーつまみが離されたか否かが判定される。そうでなければ、コントロールは判定ブロック2030に戻される。スクロールバーつまみが離されていれば、コントロールは機能ブロック2060に通され、スクロール動作を終了してシステムを非スクロール動作状態に戻り、処理はターミナル2070で完了する。

第21A図、第21B図、および第21C図は本発明によるウインドゥ・スクロールを示す。第21A図では、スクロールバーつまみ2110はウインドゥ2112の最上部にある。第21B図は、スクロールバーつまみ2120がウインドゥの中央に動かされ、従ってウインドゥの内容

2122が更新されることを示す。第21図は、スクロールバーつまみ2140がウインドゥの底部に動かされ、ウインドゥの最も底の部分が表示されていることを示している。

共同作業ロジック

(collaboration Logic)

共同作業は、プロセスを呼び出してモデル・サーバーを始動することから開始される。例えば、ユーザがディスプレイ上のドキュメントでダブルクリックすると、オペレーティング・システムのタスク開始部分が呼び出される。この呼出しにより、タスクが生成される。このタスクは、新しいタスクを生成するために必要なTTaskProgramオブジェクト・カプセル化情報によって作成される。第22図は本発明によるタスク管理のクラス階層を示している。第22図に示すブロックの各々の詳細設計は、以下で説明する。

タスク管理設計

第23図は、TTeamHandleによって別のチームに対してメイン・タスクを生成するときのプロセスを示している。TTaskProgramオブジェクトはTTeamHandle上のインターフェイスを使用して、新しいチームを作成する。”runtest-t MainTime MediaTest-l TimeMediaTest”コマンド・ラインを新しいチームが実行するとす

る。このテキストは、TCommandLineのコンストラクタ（constructor）に渡される。TTeamHandleコンストラクタはTCommandlineをフラットにし、それをストリーム化してターゲット・チーム側のナブサーバー（nubserver）へ送る。ターゲット・チーム側では、ナブサーバーはTCommandLineを復元し、TTaskProgramクラスで定義された抽象インターフェイスを使用して、”runtest”プログラムを実行する準備を行う。Initialize（初期設定）メソッドは”runtest”実行可能ライブラリを探し出し、そのライブラリとこのメソッドが必要とするライブラリをロードし、”main”のアドレスを取得する。Run（実行）メソッドは”main”をコールする。

制御の流れ（フロー）は、Initialize（初期設定）とRun（実行）の、２つの個別メソッドに分割されている。Initializeメソッドは、ユーザ・コードを実行するための全ての仕事を行う。TCommandLineサブクラス

の場合には、この作業には、必要なライブラリをロードし、”main”のエントリ・点アドレスを見つけるまでのすべてのステップが含まれる。Initializeメソッドから戻ると、TTeamHandleコンストラクタをコールしたタスクはアンブロック（unblock）される。言い換えれば、作成チーム側のコンストラクタは、ターゲット・チーム側の新しいタスクにおけるInitializeメソッドと同期している。作成タスク、”main”（argc, aegv）エントリ・ポイントのような「ユーザ・コード」に入った所まで、そのアクションが成功したとの確認が伝えられる。この場合、作成タスクは、例えば、次のことを仮定することができる。つまり、ライブラリがロードされたこと、生成タスクがエントリ・アドレスまでジャンプできること、静的データ（static data）が初期設定されたこと、メッセージを送ることができること、などである。

第２に、InitializeメソッドとRunメソッドを別々に定義すると、TTaskProgramの例外とクライアント・コードの例外とを区別する例外モデルが得られる。例えば、Initializeメソッドに例外が起こると、コンストラクタにも例外が起こって、そのアクションが失敗したことを生成チームに通知するようになっている。このようなことは、Initializeメソッドが必要とするライブラリ・セットの一部

を見つけこと、ロードすること、あるいは初期設定することに失敗した場合に起

こる。Runメソッドで例外が起こると、このイベント（事象）は、クライアント・コード中の処理されなかった例外を反映している。Runメソッドの実行中に起こった例外は、通知することやログに記録することを必要に応じて付加することができる。

付加的同期化

時には、TTaskProgramの特定サブクラスは、単純な「Initializeから戻るまでブロックする」モデルを越えて同期化を必要とするクライアントをもつ場合がある。あるタスクの生成タスク（creator）が、ターゲット・タスクがある既知の状態に達するまでそのターゲット・タスクと同期をとる必要があるとする。以下では、単純なプロトコルについて説明する。生成タスクはやりとりにおいてTTaskProgramに渡され、そこで生成タスク、または他のエンティティはあとでreveive（受信）を実行する。ターゲット・タスクがある事前に決めた状態に達すると、ターゲット・タスクは受け取ったやりとりに対しsend（送信）を実行する。この応答はターゲット・タスクをアンブロックするので、生成タスク（またはやりとりを知っている他のエンティティ）が、タスクが事前に決めた状態に達したことを確認したことがターゲット・タスクに知らされる。第24図は、本発明による詳細ロジックを示すフローチャートである。処理はターミナル2400から

開始され、機能ブロック2410に制御がこのターミナルから即時に渡され、ここでユーザがドキュメント・オブジェクトに対してダブルクリックする。そのあと、機能ブロック2420で、タスク・プログラムは新しいアドレス空間（address space）を生成し、オブジェクトをこのアドレス空間に挿入し、タスクをアドレス空間に生成させ、そのオブジェクトのメソッドを開始させる。機能ブロック2430に示すように、このオブジェクトはドキュメント・ファイルをオープンし、ドキュメント・オブジェクトを復元し、オブジェクトのスタート（start）メソッドをコールする。このメソッドは、まず、モデル・サーバーがそのドキュメントに対してアクティブであるかどうかを調べる。次に、モデル・サーバーがアクティブ

でなければ、メソッドは機能ブロック2440　に示すようにモデル・サーバーを生成して開始し、ドキュメントを読み込み、機能ブロック2450に示すようにドキュメントに関連するユーザ・インターフェイスをオープンする。この時点で、ユーザはコマンドを入力することができる。モデル・サーバーがあれば、メソッドは既存のモデル・サーバーと接続し、ドキュメントのコピーをモデル・サーバーから取り出し、ユーザ・インタフェイスをオープンし、この時点で、ユーザはコマンドを入力することができる。

コマンドには、次の２つの基本タイプがある。非反

復コマンドと反復コマンドである。非反復コマンドはそのアクションを単一のアトミック・オペレーション（atomic operation）として実行する。反復コマンドは開始フェーズ、ゼロまたは１つ以上の継続フェーズおよび終了フェーズをもっている。ある種のコマンドでは、継続フェーズのいくつかは、コマンドの結果に影響を与えることなくスキップすることができる。非反復コマンドでは、トラッカ（tracker）は、ユーザがなんらかのアクションを行うと呼び出されて、機能ブロック2460に示すようにユーザのアクションを調べ、ユーザがどのコマンドを出そうとしているかを判断し、そのコマンドのdo（実行）メソッドを実行する。例えば、ユーザがアイコンでダブルクリックすると、ダブルクリックされたアイコンに対してオープン・コマンドが出される。

コマンドのdoメソッドはコマンドをモデル・コマンド・ディスパッチャ（model command dispatcher）に送る。モデル・コマンド・ディスパッチャ（MCD）はコマンドを受け取り、機能ブロック2470に示すようにコマンドをモデル・サーバーに配布する。モデル・サーバーは、共同作業者（collaborator）のリストを維持し、どの共同作業者がモデルを変更する権限を持っているかを決定することを受け持つオブジェクトであり、機能ブロック2480に示すようにコマンドをすべてのアクティブな共同作業者に配布する。

モデル・サーバーはコマンドを受け取ると、コマンドを送った共同作業者がコマンドを出すことが許されているかどうかを判断する。許されていなければ、エ

ラーが返される。コマンドが許されていれば、コマンドは、他のアクティブな共同作業者のモデル・コマンド・エグゼクティブ（executive）の各々に送られ、コマンドは、コマンドを送った協同作業者のモデル・コマンド・ディスパッチャに返却される。モデル・コマンド・ディスパッチャはリターン・コマンドを元のコマンドに格納してから、コマンドのhandle do（実行処理）メソッドを実行する。このメソッドはコマンドに基づいてモデルを修正する。モデルは通知を生成し、この通知は通知フレームワークによって、関係するビューに配布される。ビューは通知を受け取り、通知とモデルを調べて、モデルの現在のビューを表示する。例えば、あるオブジェクトをダブルクリックすると、選択されたオブジェクトをREDに変えるコマンドの場合は、ビューは、そのオブジェクトでダブルクリックされると、オブジェクトをREDオブジェクトとして再描画することになる。そのあと、グラフィック・ビューはオブジェクトをカラー・レッド・オブジェクトとして再描画する。これに対し、テキストのみのディスプレイは、オブジェクト上にラベルREDを入れる。通知が生成されたあと、制御はディスパッチャに戻され、ディスパッチャから制御がトラッカに

戻される。

共同作業者のモデル・コマンド・エグゼクティブはモデル・サーバーからコマンドを受け取ると、コマンドのhandle doメソッドをコールする。このメソッドはコマンドに基づいてモデルを修正する。モデルは通知を生成し、この通知は通知フレームワークによって関係するビューに配布される。ビューは通知を受け取り、通知とモデルを調べて、モデルの現在のビューを表示する。例えば、あるオブジェクトでダブルクリックすると、選択されたオブジェクトをREDに変えるコマンドの場合は、ビューは、そのオブジェクトでダブルクリックされると、オブジェクトをREDオブジェクトとして再描画することになる。そのあと、グラフィック・ビューはオブジェクトをカラー・レッド・オブジェクトとして再描画する。これに対し、テキストのみのディスプレイは、オブジェクト上にラベルREDを入れる。通知が生成されたあと、制御はモデル・コマンド・エグゼクティブに戻り、別のコマンドを待つ。

モデル・ベース・トラッキングの詳細

はじめに

単純なトラッキング

第25図は、アップル・マッキントッシュなどの従来システムで使用されている代表的なトラッキング（追跡）ループを示す図である。このループによると、ユーザはマウスおよびスクリーンを使用してコンピュータと対話（やりとりすること）することができる。

単純なトラッキングの問題

単純なトラッキングは、様々な種類のユーザのやりとりでは十分な働きをしているが、ドキュメント・アーキテチャで使用するには困難である。ドキュメント・アーキテクチャによると、同一データの多重ビューを複数のマシン（計算機）に置いておくことができる。可能な限りすべての共同作業者のマシンに置かれている可能な限りすべての種類のビューに対してフィードバックをどのように描画するかを各トラッカが知っていることを要求するのは大変である。

抽象トラッカとフィードバッカ

抽象トラッカは、多重ビューにおいておよび複数の

マシンに共通してトラッキングをサポートするために使用された最初の試みである。トラッキングが開始されると、１つの抽象トラッカと複数のフィードバッカが生成される。フィードバッカは、各マシン上の各ビューごとに１つが生成される。抽象トラッカは具体的なデバイス・ベースのイベントを抽象モデル・ベースのイベントに変換する。モデル・イベントは種々のフィードバッカに配布され、そこでモデル・イベントはビューにフィーバックを送るために使用される。第26図は、抽象トラッカ・ループの例を示す図である。抽象トラッカによると、協同作業による複数ビューのトラッキングが可能であるが、次のような問題がある。

・　フィードバッカは、すでにビューに取り込まれている表示コードを重複して作ることで終わるのが通常である。

・　フィードバッカは、モデルを増分的に修正するように見せかけているだけである。テキスト・エディタや拘束ベースの3Dグラフィック・エディタのような

複雑なモデルでは、フィードバッカは、ユーザの入力の効果を十分にシミュレートできない場合がある。

モデル・ベース・トラッキング

トラッカが実際にモデルを修正するようにすれば、

すべてがはるかに簡単になる。事実、これがモデル・ベースのトラッキング・ループで行われていることである。トラッカがモデルに対してコマンドを出すと、モデルは、第27図に示すように、すべての関係するビューに変更通知を知らせる。下に示したのは、トラッカを実現するために使用されたＣ言語コードである。

```
class TMyTracker :: public TTracker
{
public:
                                    TMyTracker(TMyModel* model);
    virtual                        ~TMyTracker();
    virtual TTracker*  TrackFirstTime(const TEvent&);
    virtual TTracker*  TrackContinue(const TEvent&);
    virtual void            TrackLastTime(const TEvent&);
private:
    TMyModel*                fMyModel;
    TMyCommand                   fMyCommand;
};

TMyTracker::TMyTracker(TMyModel* model)
{
    fModel = model;
}

TTracker* TMyTracker::TrackFirstTime(const TEvent& event)
{
    fMyCommand.SetData(((const TMyEvent&) event)->GetSomeData());
    fMyModel->ProcessDoFirstTime(fMyCommand);
    return this;
}

TTracker* TMyTracker::TrackContinue(const TEvent& event)
{
    fMyCommand.SetData(((const TMyEvent&) event)->GetSomeData());
    fMyModel->ProcessDoContinue(fMyCommand);
    return this;
}

void TMyTracker::TrackLastTime(const TEvent&)
{
    fMyCommand.SetData(((const TMyEvent&) event)->GetSomeData());
    fMyModel->ProcessDoLastTime(fMyCommand);
}
```

コマンド

コマンドの仕事は、モデルを増分的に修正することである。コマンドは一度だ

け実行されるのではなく、増分的に実行される。また、コマンドは一度だけストリーミングされるのではなく、コマンド・デルタ・オブジェクトで更新される。コマンドを更新するために使用されるメソッドには、２組がある。下記に呼出し順序を示す。

```
virtual Boolean     StreamOutContinueDelta(TStream&) const;
virtual void        StreamInContinueDelta(TStream&);
virtual void        StreamOutLastDelta(TStream&) const;
virtual void        StreamInLastDelta(TStream&);
```

StreamOut...Deltaメソッドを書くときのルールはこのトラック・フェーズ期間に変更されたデータをストリーム・アウトすることである。

```
Boolean TMyCommand::StreamOutContinueDelta(TStream& towhere)
const
{
    fData >>= toWhere;
    return FALSE;
}
```

StreamOutContinueDeltaメソッドは、このデルタが必要であれば、TRUE（真）を返却する。ラバー・バンド・ライン（rubber-band-line）トラッカのような、ある種のトラッカは、その中間ステップの一部または全部をスキップすることができる。この種のトラッカは

常にStreamOutContinueDeltaからFALSE（偽）を返却するようになっている。ポリゴン・スケッチング・トラッカ（polygon sketching tracker）のような、他の種類のトラッカは、トラックのドラグ部分の間にFALSEを返却するが、頂点がクリックされるたびにTRUEを返却する。

StreamIn...Deltaメソッドを書くときのルールは、対応するStreamOut...Deltaメソッドによってストリーム・アウトされたデータをストリーム・インすることである。

```
void TMyCommand::StreamInContinueDelta(TStream& fromWhere)
{
    fData <<= fromWhere;
}
```

多くのコマンドは、トラッキングの各フェーズ期間に正確に同じ情報を受け渡す必要がある。これらのコマンドの書き方を単純化するために、Stream...Deltaメソッドのデフォルトは、次のようになっている。

```
Boolean TCommand::StreamOutContinueDelta(TStream& stream) const
{
    *this >>= stream;
    return FALSE; // 継続デルタは省略時値では必須でない
}

    void TCommand::StreamInContinueDelta(TStream& stream)
    {
        *this <<= stream;
    }
    void TCommand::StreamOutLastDelta(TStream& stream) const
    {
        StreamOutContinueDelta(stream);
    }
    /void TCommand::StreamInLastDelta(TStream& stream)
    {
        StreamOutContinueDelta(stream);
    }
```

これらの省略時メソッドが用意されていると、operator>>=とoperator<<=をオーバーライドして、すべての3トラック・フェーズ期間にストリーミングを行うことができる。トラッカがTModel::ProcessDoFirstTime()をコールするとき、コマンド引数はフラットにされて、モデル・サーバーへ送られる。そこから、引数は再びフラットにされて、キャッシュ・モデル（cached model）の各々へ送られる。各モデルでは、コマンドのHandleDoFirstTime( )メソッドが実行される。

トラッカがTModel::ProcessDoContinue()をコールするときは、コマンド引数は、デルタ情報をストリーム・アウトするように要求される。コマンド・デルタ

はモデル・サーバーへ送られる。そこから、デルタはキャッシュ・モデルの各々へ送られる。各モデルで

は、デルタはコマンドのローカル・コピーにストリーム・インされる。そのあと、コマンドのHandleDoContinue()メソッドが実行される。

トラッカがTModel::ProcessDoLastTime()をコールするときは、コマンド引数は、デルタ情報をストリーム・アウトするように要求される。コマンド・デルタはモデル・サーバーへ送られる。そこから、デルタはキャッシュ・モデルの各々へ送られる。各モデルでは、デルタはコマンドのローカル・コピーにストリーム・インされる。そのあと、コマンドのHandleDoLastTime()メソッドが実行される。

増分（incremental）コマンドがその拡張Doフェーズを終了できる方法には、２つある。標準的方法は、TModel::ProcessDoLastTime()がコールされる場合である。もう１つの方法は、トラッキングを行っている共同作業者が予期しないで共同作業から去る場合である。その場合は、モデル・サーバー側のコマンドは、そのHandleCollaboratorDied()メソッドをコールさせる。

拡張Doフェーズが終了したあと、コマンドは、そのHandleDo()メソッドがコールされたのと同じ状態にあるものと期待される。モデル・サーバー側のコマンドは、ログ記録とアンドゥ（取消し）目的のためにコマンド・マネージャに渡される。キャッシュ・モデル側のコマンドは削除される。

デフォルトでは、コマンドは、ローカル・キャッシュ・モデルに適用される前にモデル・サーバーへ送られる。これにより、HandleModelServerDo()メソッドには、モデル・サーバーとやりとりする機会が与えられる。これは、ある種のコマンド（カットやペーストおよびオブジェクト生成コマンドなど）を実現することを大幅に単純化するが、往復時間による遅延でり、大部分のトラッキング・コマンドが遅くなる。コマンドがモデル・サーバーの状態に依存していなければ、AllowsEarlyDoをオーバーライドし、TRUEを返却することにより、コマンドのローカル実行を高速化することができる。これにより、ドキュメント・アーキテク

チャは、コマンドを即時に実行してから、コマンドをモデル・サーバーへ送ることができる。

Virtual Boolean AllowsEarlyDo()const；

AllowsEarlyDo()は、トラッキングの継続および最終フェーズを含めて、コマンドが実行されるたびにチェックされる。このメソッドがコールされるたびに、異なる答を返すことができる。

コマンドによっては、終了条件が明確に定義されていないものがある。例えば、テキスト入力トラッキングは、別のコマンドが開始されたときだけ終了する。この種のコマンドをサポートするために、モデルは、別のコマンドが処理されたとき、保留中の増分コマンドを自動的に終了するようになっている。モ

デルは、現在のトラッキング・コマンドのTCommand::FinishTr ackingメソッドをコールすることによって、これを行う。

Virtual Void TCommand::FinishTracking()；

Finish Trackingのデフォルトの挙動は次のようになっている。

```
void TCommand::FinishTracking()
    {
    GetModel()->ProcessDoLastTime(*this);
    GetTracker()->StopTracking();
    }
```

注意すべきことは、大部分のトラッカは、StopTrackingメソッドがコールされたとき、自身（およびそのコマンド）を削除することである。このことは、Stop Trackingをコールしたあと、インスタンス変数のどれも参照してはならないこと（またはどのバーチャル・メソッドもコールしてはならないこと）を意味する。

この挙動をサポートするために、SetTrackerメソッドをコールすることによって、トラッキング・コマンドにそのトラッカについて知らせる必要がある。SetTrackerは、コマンドの構築中に呼び出すことができるが、コマンドがトラッカによって初めて実行される前ならば、いつでも呼び出すことも可能である。

モデル

モデルの仕事は、データのレポジトリになること、およびデータが修正されたとき変更通知を知らせることである。変更イベントが起こると、ビューがインテリジェントに自身を更新できるだけの、変更に関する十分な情報が得られる。これには、通常、選択の変更前の旧値が含まれている。約束により、変更通知はデータが修正された後送られる。

ビュー

ビューの仕事は、モデル内のデータを表示することおよび変更通知に即時に反応することである。データを十分に迅速に更新できれば、変更通知に対する最も単純な応答の１つは、TView::InvalidateALL()を実行することである。通常のビューでは、モデル変更通知に対する迅速な応答を念頭に置いてビューを設計する必要がある。これは、見かけほど難しくない。描画プログラムの場合は、２つのオフ・スクリーン・バッファ（off-screen buffer）を採用することが、すぐれた設計である。最初のバッファには、選択されなかったオブジェクトがすべて置かれることになる。２番目のバッファは、選択されたオブジェクトを選択されなかったオブジェクトの上に合成するために使用されることになる。そのあと、２番目のバッファはスクリー

ンを更新するために使用されることになる。

特定のシステム環境における好適実施例を中心に本発明を説明してきたが、本発明は、変更を加えることにより、請求の範囲に記載された本発明の精神と範囲内で、他の異なるハードウェアおよびソフトウェア環境で実施することができることはもちろんである。

【図1】

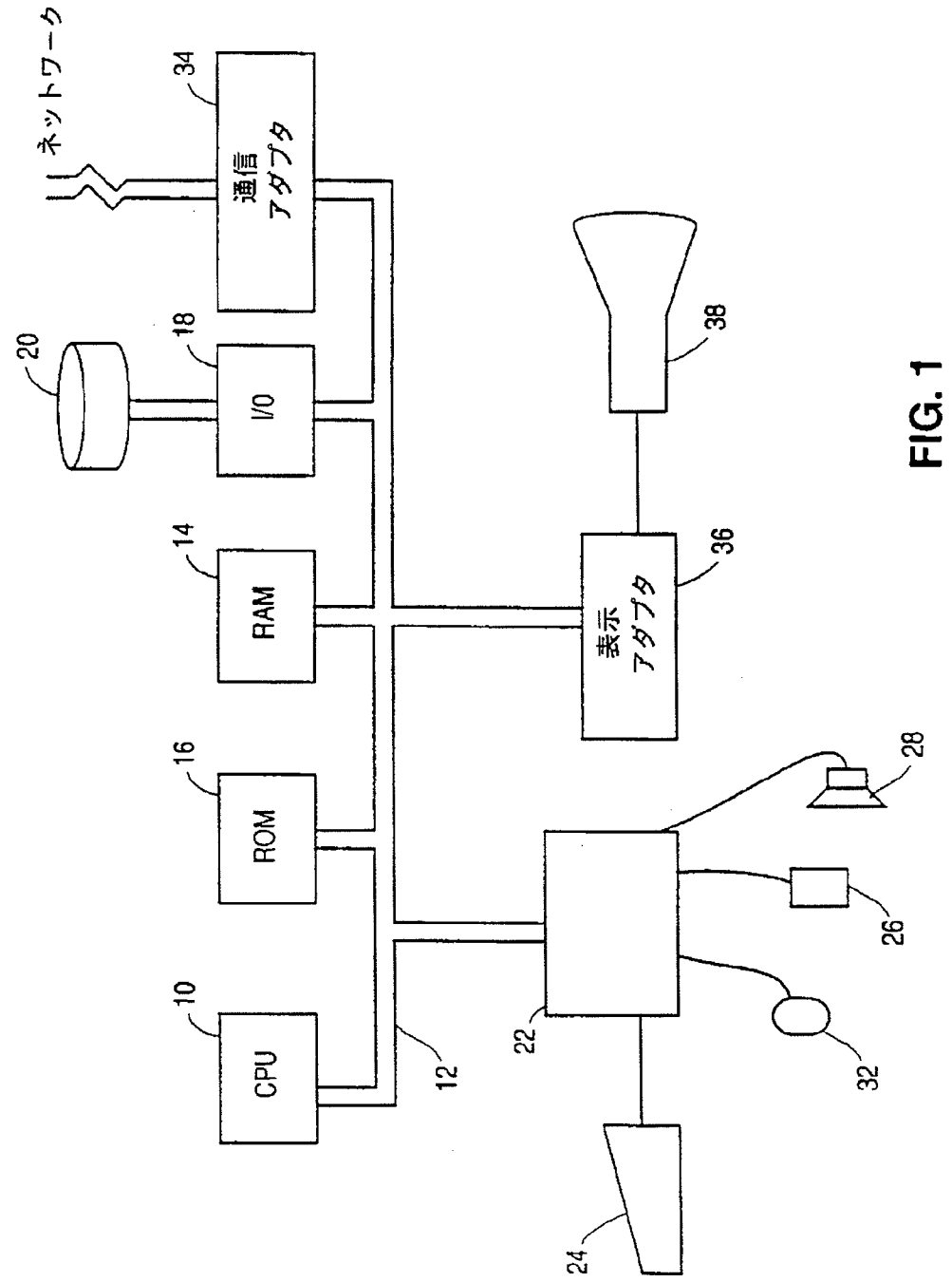

ネットワーク
34
通信
アダプタ
18
I/O
20
14
RAM
16
ROM
10
CPU
12
38
36
表示
アダプタ
28
26
22
32
24

FIG. 1

【図１Ｂ】

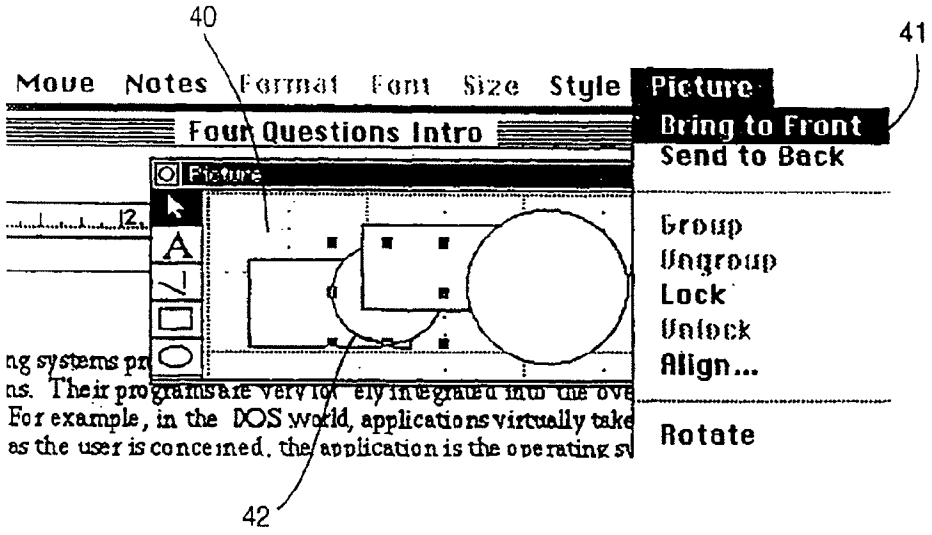


FIG. 1B

【図２】

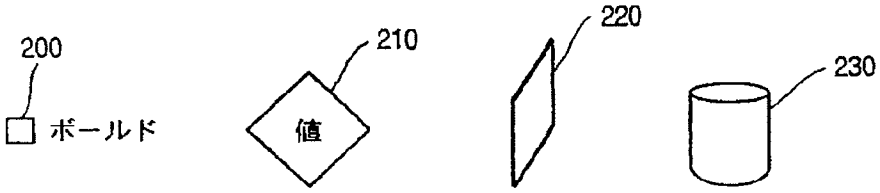


FIG. 2

【図３】

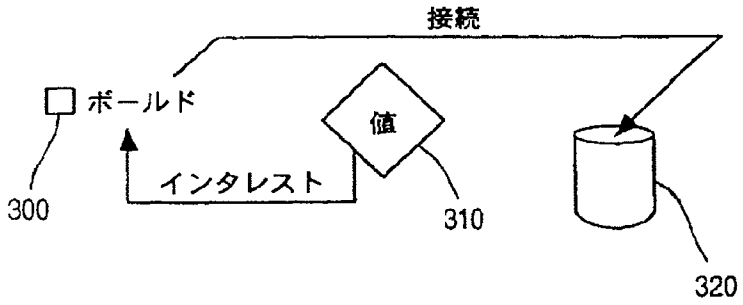


FIG. 3

【図４】

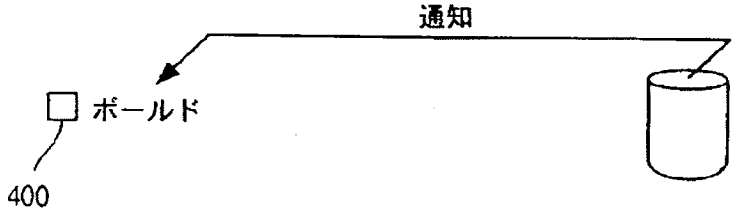


FIG. 4

【図５】

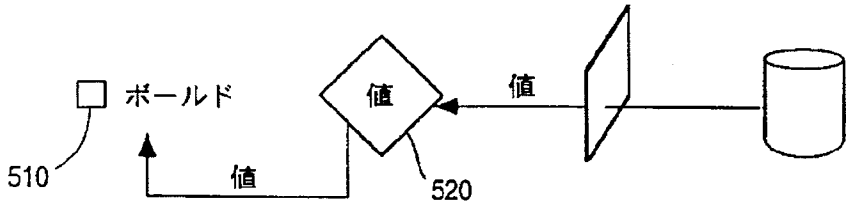


FIG. 5

【図６】

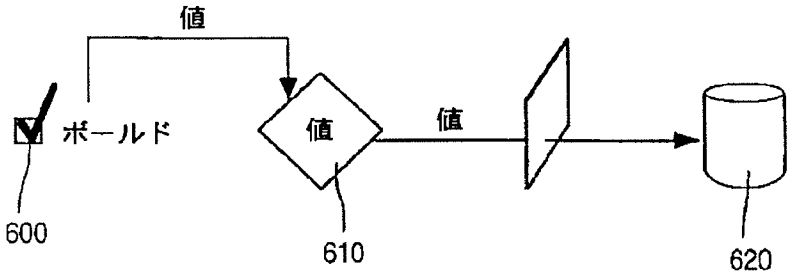


FIG. 6

【図７】

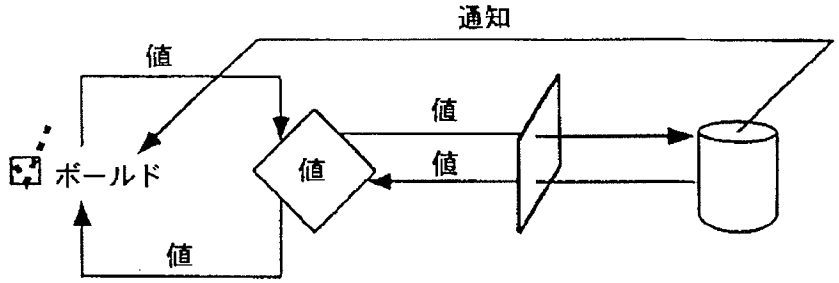


FIG. 7

【図８】

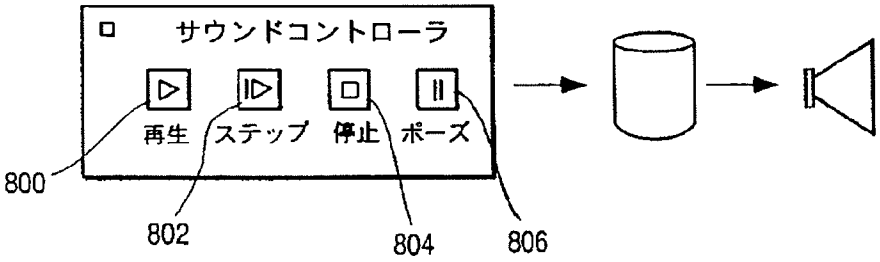


FIG. 8

【図９】

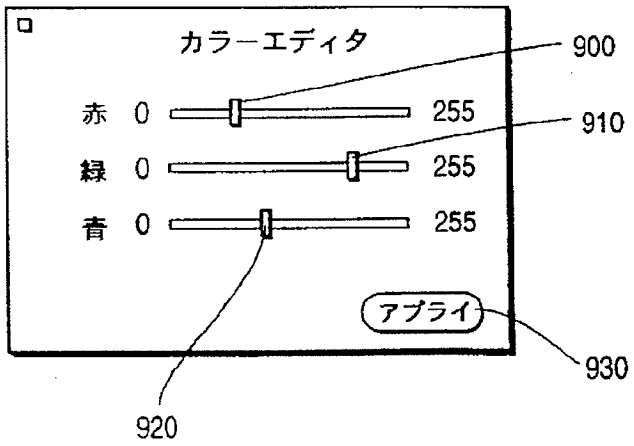


FIG. 9

【図１０】

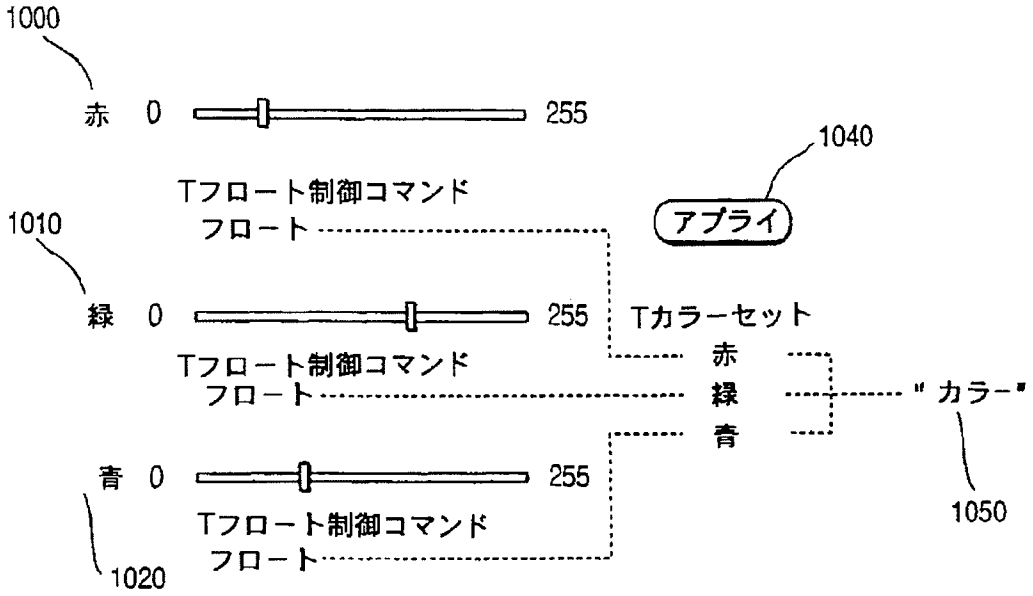


FIG. 10

【図１１】

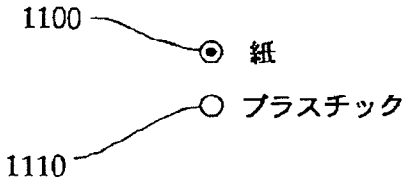


FIG. 11

【図１２】

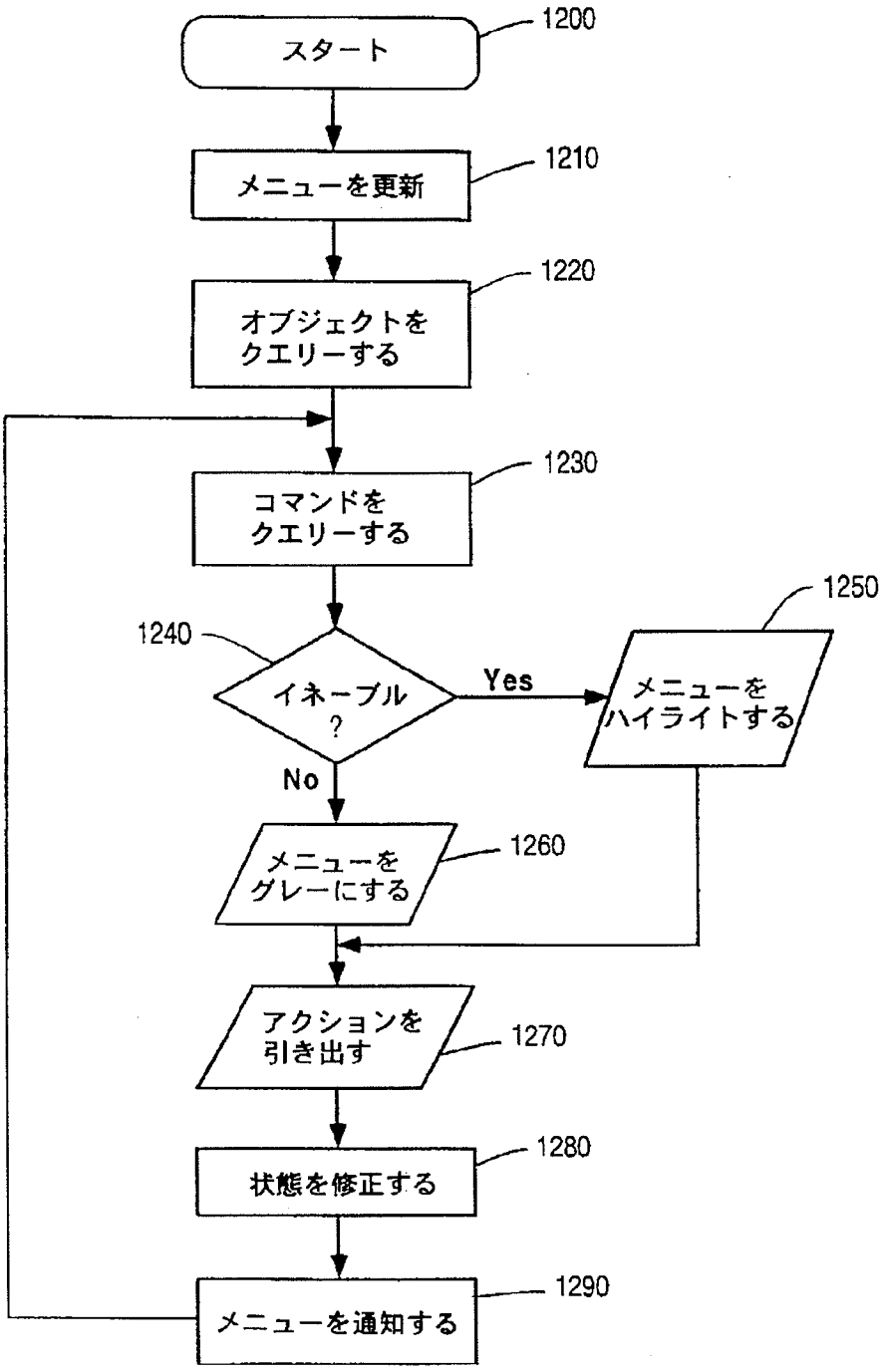

スタート
1200
メニューを更新
1210
オブジェクトを
クエリーする
1220
コマンドを
クエリーする
1230
1240
イネーブル
?
Yes
1250
メニューを
ハイライトする
No
メニューを
グレーにする
1260
アクションを
引き出す
1270
状態を修正する
1280
メニューを通知する
1290

FIG. 12

【図１３】

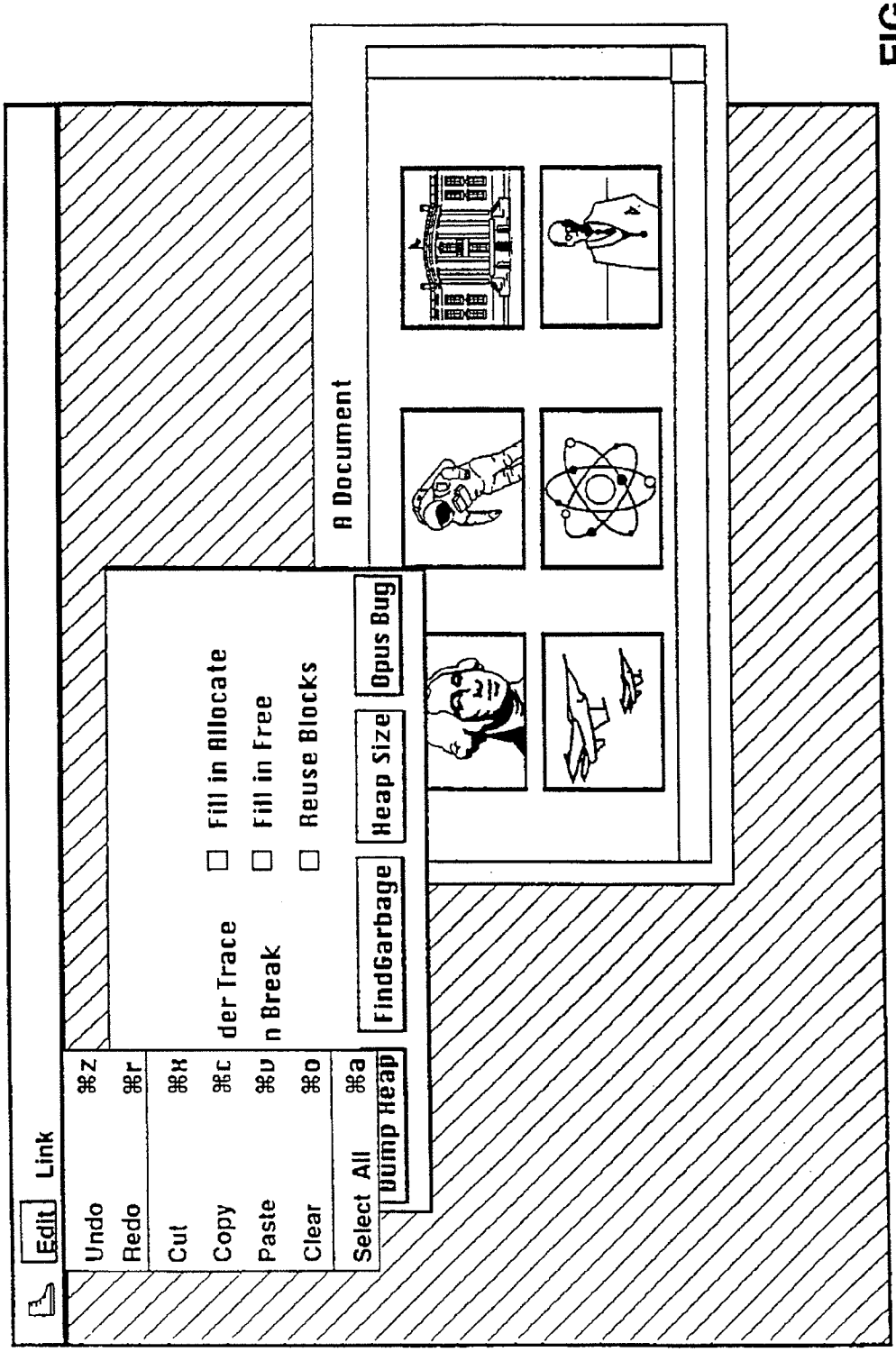

Edit
Link
Undo ⌘z
Redo ⌘r
Cut ⌘x
Copy ⌘c
Paste ⌘v
Clear ⌘o
Select All ⌘a
Dump Heap
der Trace
n Break
Fill in Allocate
Fill in Free
Reuse Blocks
FindGarbage
Heap Size
Opus Bug
A Document

FIG. 13

【図１４】

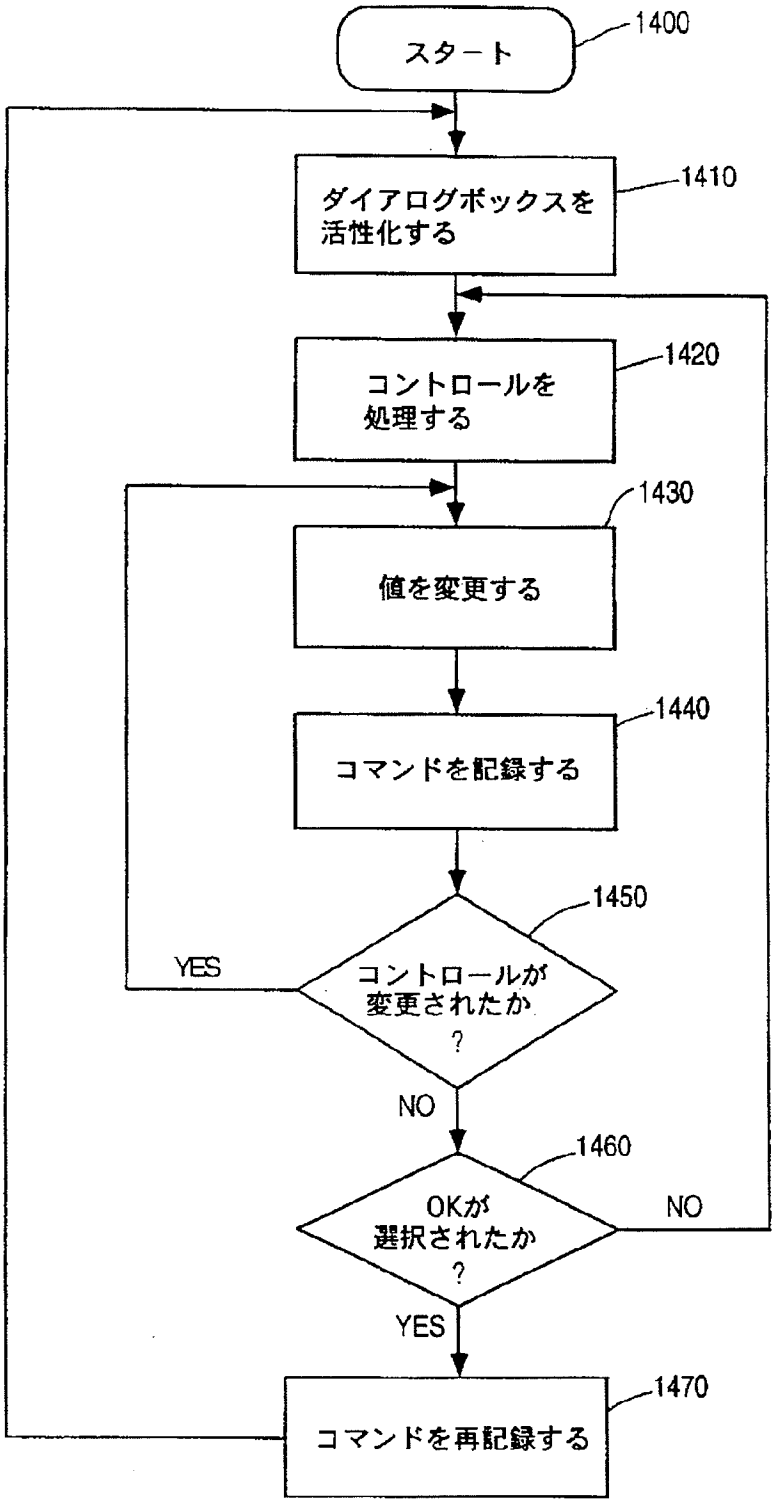

スタート
1400
ダイアログボックスを活性化する
1410
コントロールを処理する
1420
値を変更する
1430
コマンドを記録する
1440
コントロールが変更されたか？
1450
YES
NO
OKが選択されたか？
1460
NO
YES
コマンドを再記録する
1470

FIG. 14

【図１５】

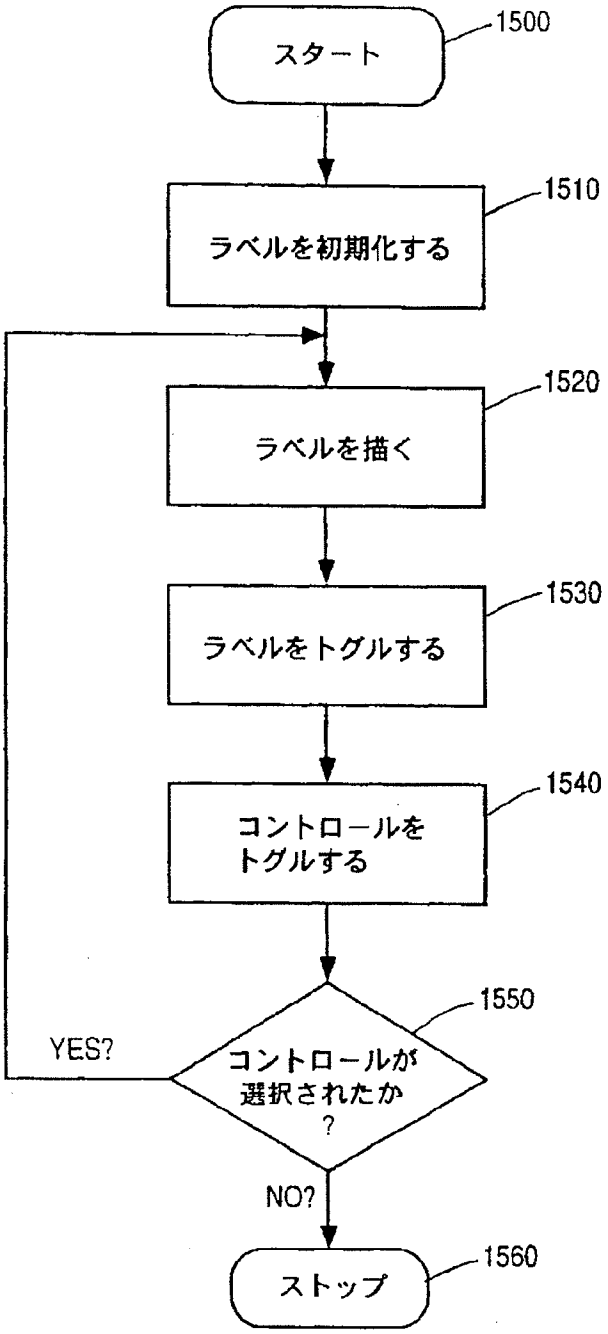

スタート
1500
ラベルを初期化する
1510
ラベルを描く
1520
ラベルをトグルする
1530
コントロールを
トグルする
1540
コントロールが
選択されたか
?
1550
YES?
NO?
ストップ
1560

FIG. 15

【図１６】

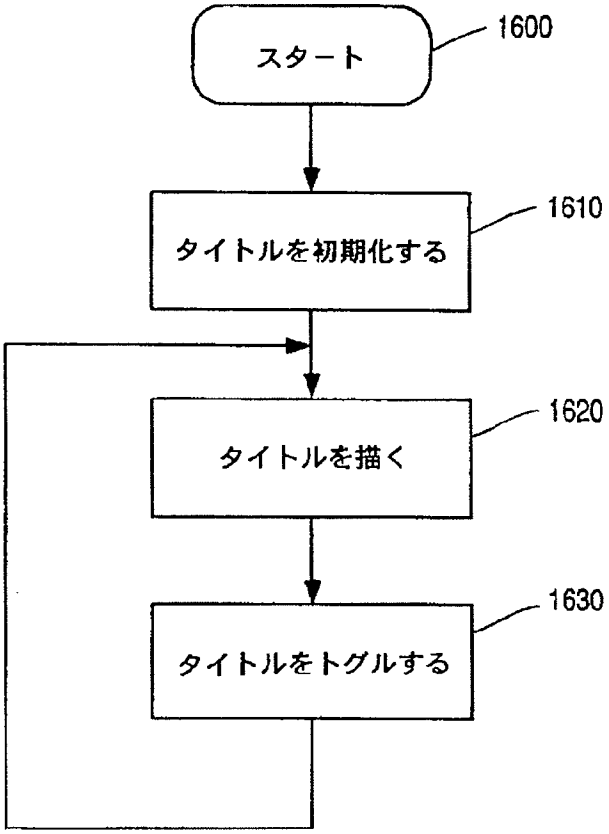


FIG. 16

【図１７】

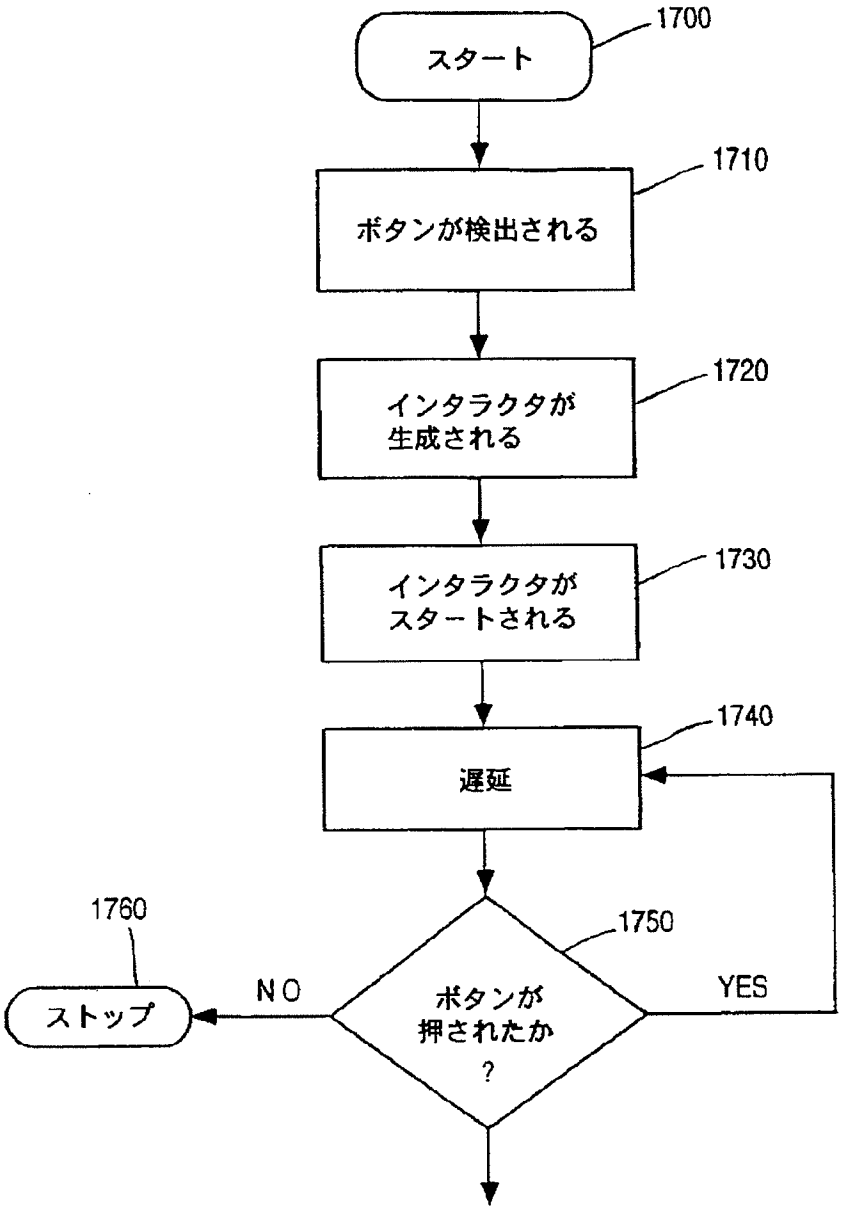

スタート
1700
ボタンが検出される
1710
インタラクタが
生成される
1720
インタラクタが
スタートされる
1730
遅延
1740
ボタンが
押されたか
？
1750
NO
YES
ストップ
1760

FIG. 17

【図１８】

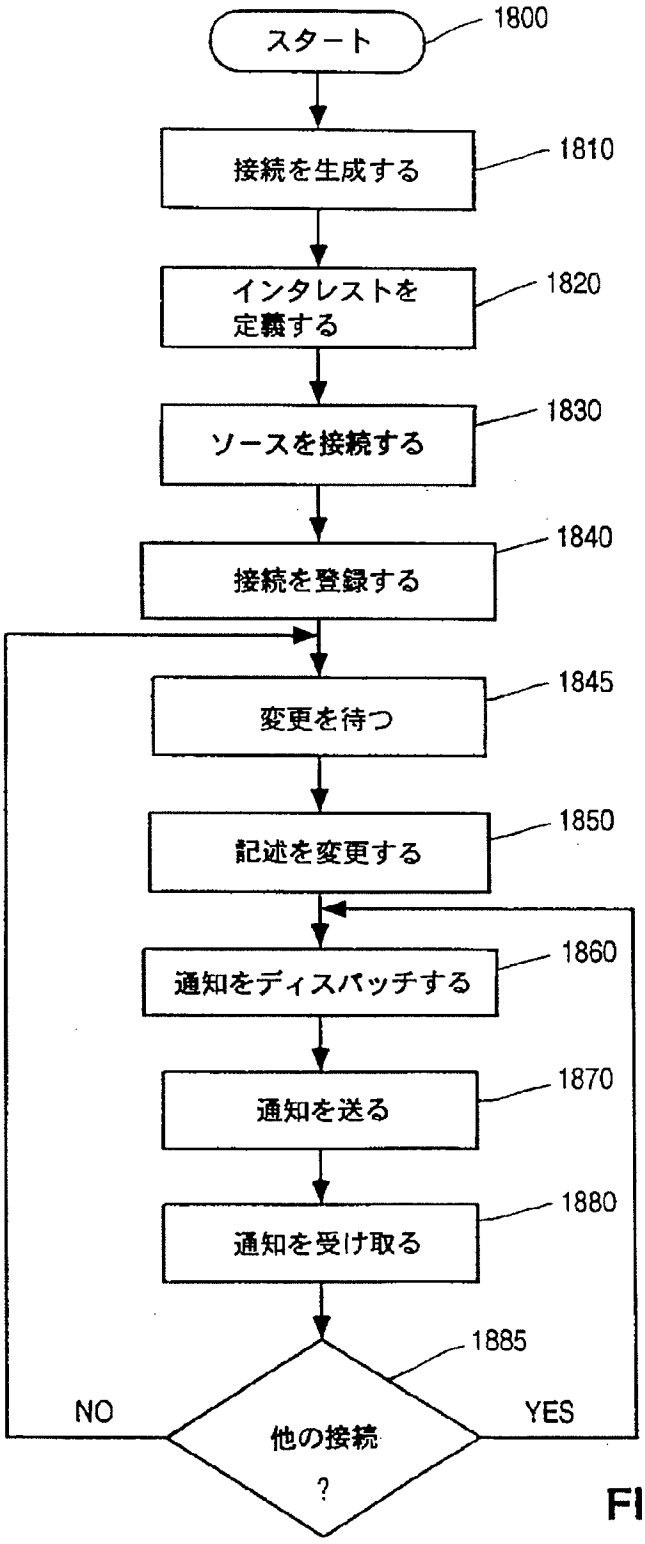

スタート
1800
接続を生成する
1810
インタレストを
定義する
1820
ソースを接続する
1830
接続を登録する
1840
変更を待つ
1845
記述を変更する
1850
通知をディスパッチする
1860
通知を送る
1870
通知を受け取る
1880
1885
他の接続
?
NO
YES

FIG. 18

【図１９】

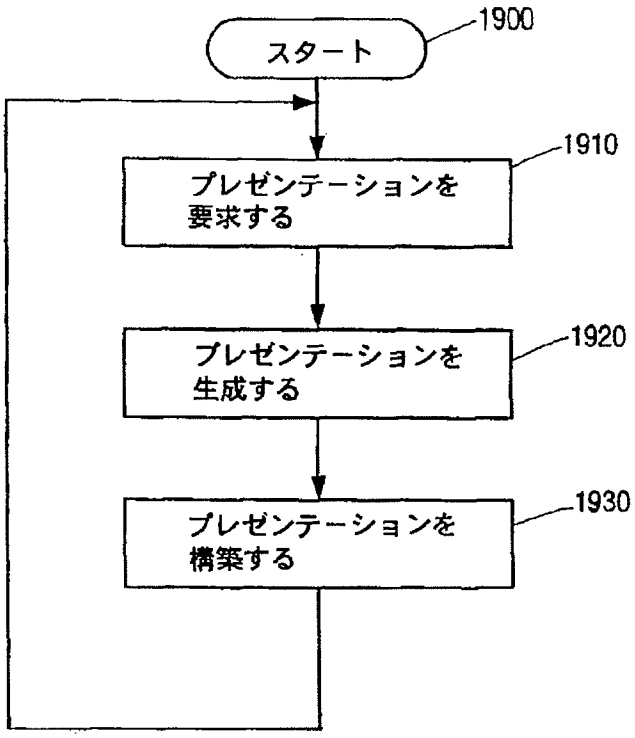

スタート
1900
プレゼンテーションを
要求する
1910
プレゼンテーションを
生成する
1920
プレゼンテーションを
構築する
1930

FIG. 19

【図２０】

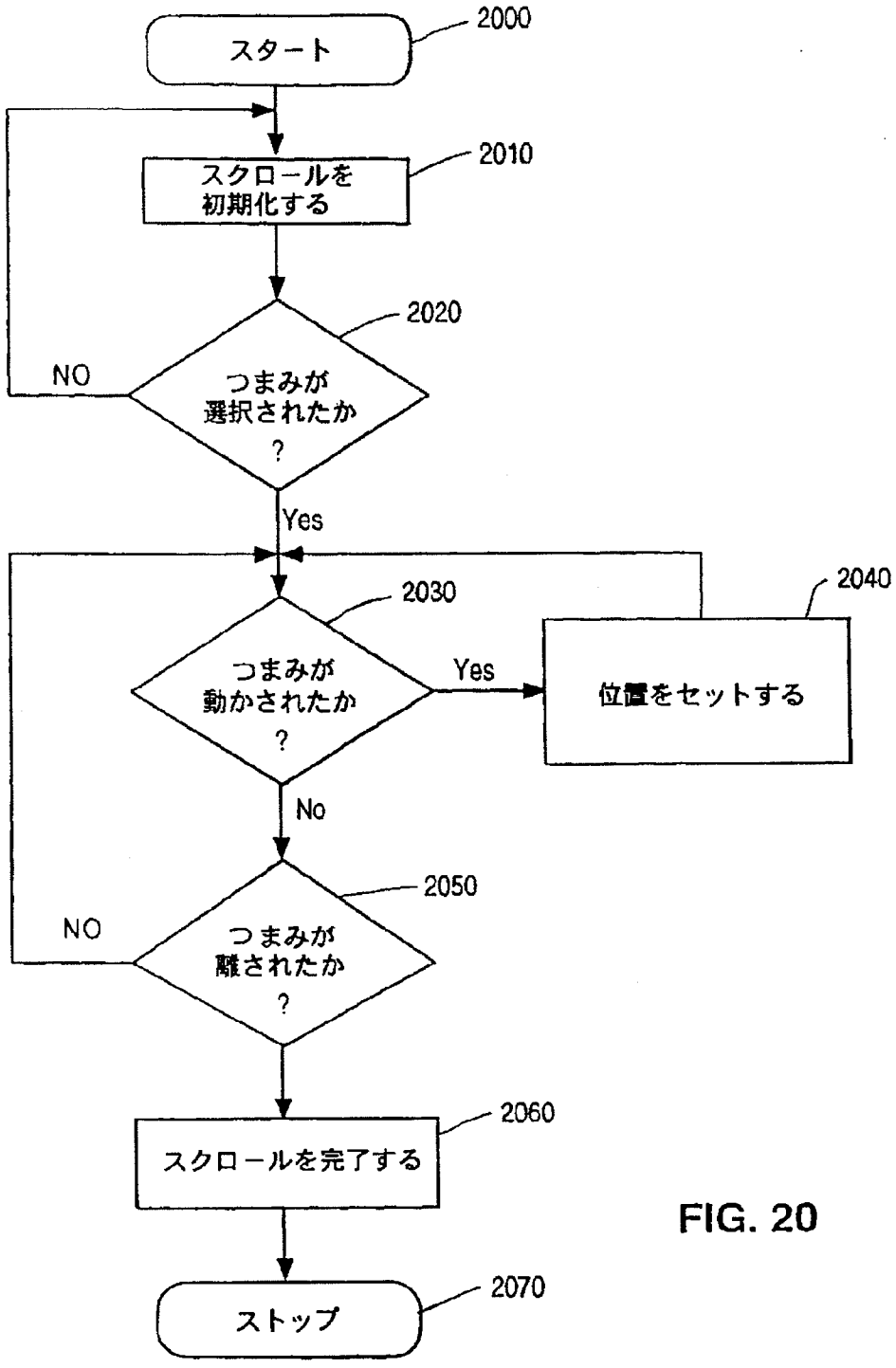

スタート
2000
スクロールを
初期化する
2010
2020
つまみが
選択されたか
?
NO
Yes
2030
つまみが
動かされたか
?
Yes
2040
位置をセットする
No
2050
つまみが
離されたか
?
NO
2060
スクロールを完了する
ストップ
2070

FIG. 20

【図２１】

FIG. 21A

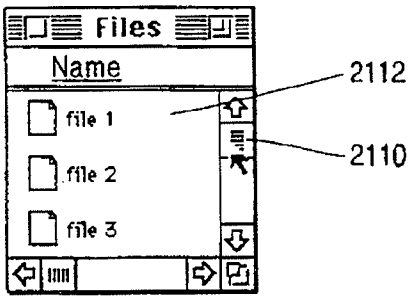


FIG. 21B

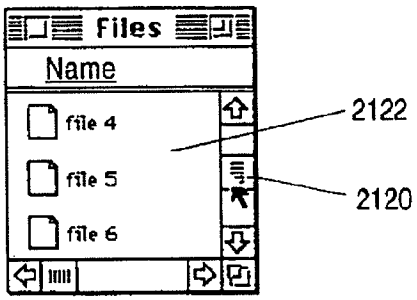
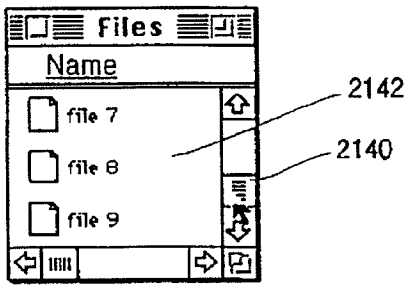


FIG. 21C



【図２２】

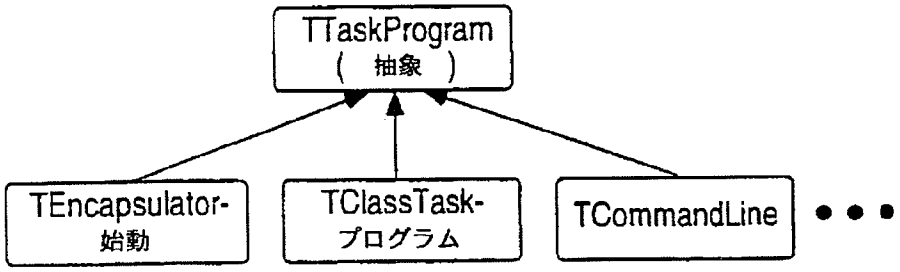


FIG. 22

【図２３】

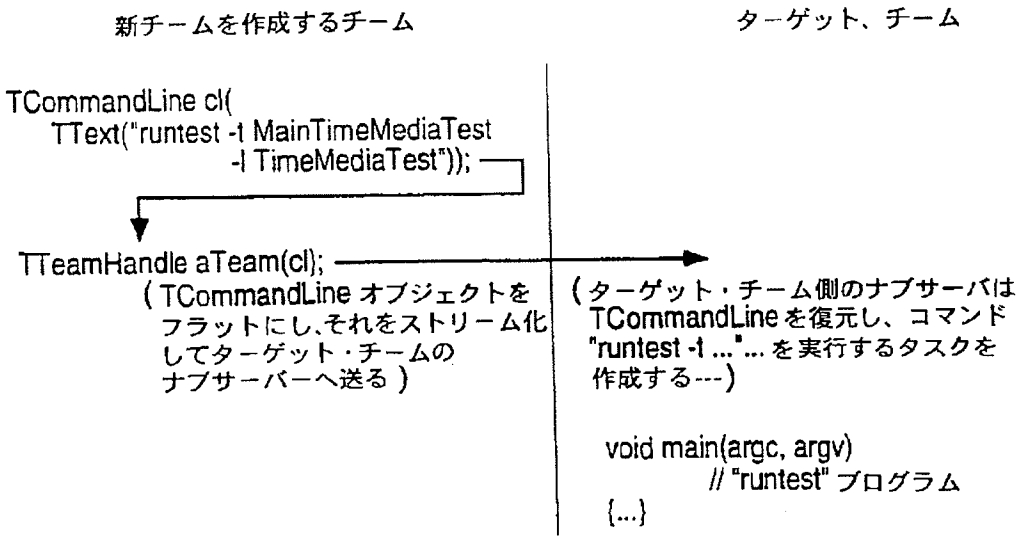


FIG. 23

【図２４】

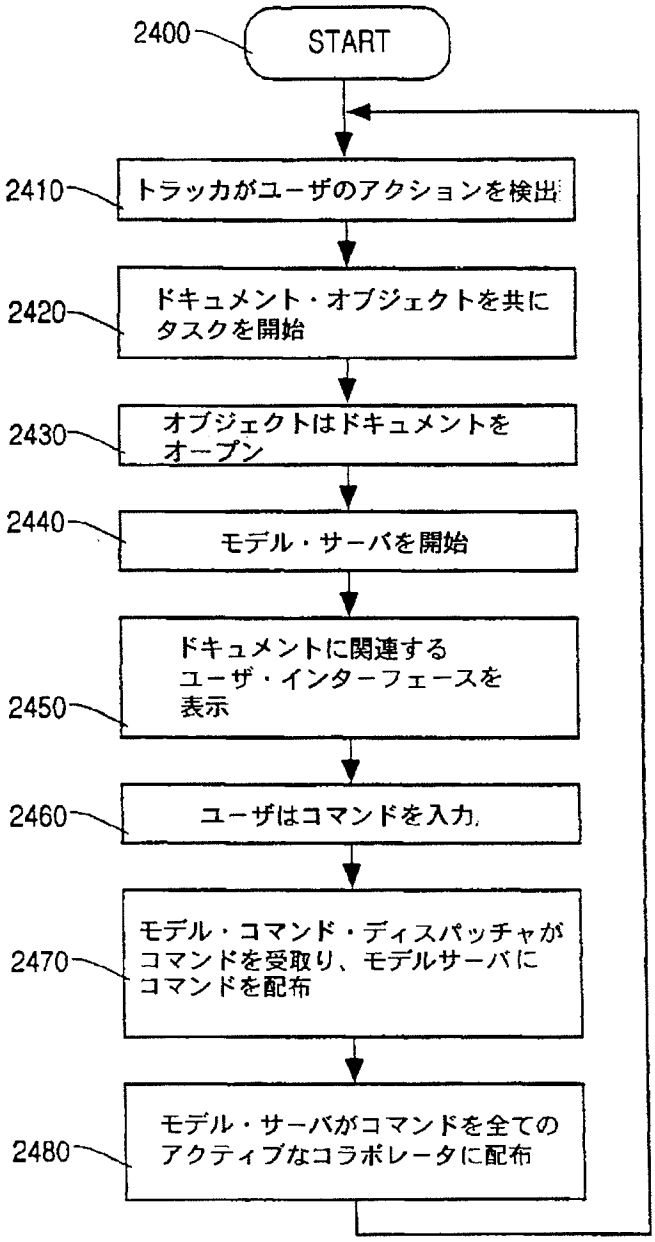
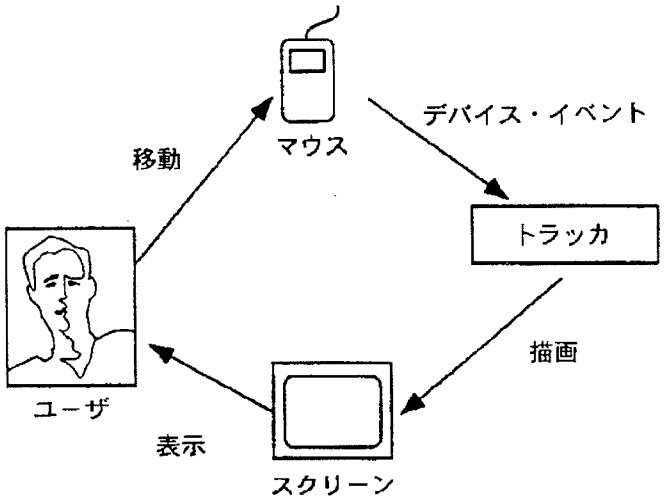

2400
START
2410
トラッカがユーザのアクションを検出
2420
ドキュメント・オブジェクトを共にタスクを開始
2430
オブジェクトはドキュメントをオープン
2440
モデル・サーバを開始
2450
ドキュメントに関連するユーザ・インターフェースを表示
2460
ユーザはコマンドを入力
2470
モデル・コマンド・ディスパッチャがコマンドを受取り、モデルサーバにコマンドを配布
2480
モデル・サーバがコマンドを全てのアクティブなコラボレータに配布

FIG. 24

【図２５】



従来技術

FIG. 25

【図２６】

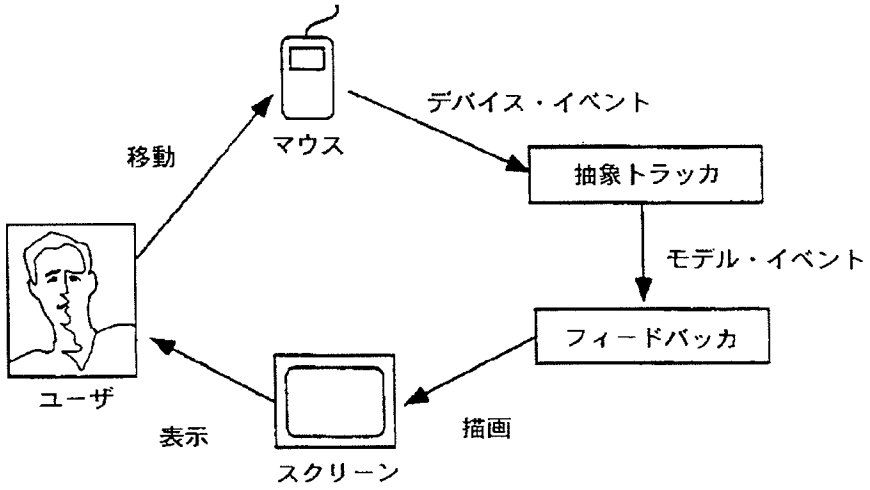


FIG. 26

【図２７】

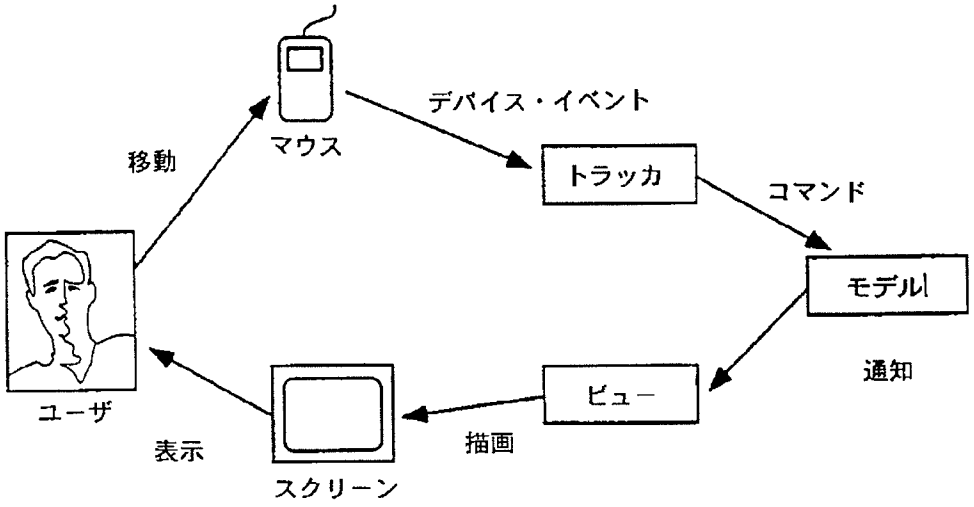


FIG. 27

【手続補正書】特許法第１８４条の８

【提出日】１９９５年３月３日

【補正内容】

（原文明細書第１頁）

明細書

並行フレームワーク・システム

著作権表記

本特許出願は、著作権保護の対象となる内容を含んでいる。著作権者は何人が情報を得る目的で特許出願の一部として複製してもそれを妨げるものではないが、他のすべての権利、特に本内容の商業的利用に関する権利を留保する。

発明の分野

本発明は、一般的には、コンピュータ・ドキュメントに関し、特にフレームワークの並行使用に関する。

背景技術

ワークステーション・ソフトウェアの開発者の間で、ユーザ・インターフェイス内の一貫性を維持しながらフレキシブルなソフトウェア環境を提供することの重要性が増してきている。この種の動作環境を提供する初期の試みがヘルナンデスらの米国特許第4,686,522号に開示されている。この特許はカーソルの位置にある動的メニューをユーザが呼び出して、そのメニューから種々の機能を呼び出すことができるグラフィックとテキストの結合処理システムについて議論している。この種のユーザとの自然な相互作用あるいは対話はユーザ・インターフェイスを改善し、アプリケーションをより直感的にする。

印刷物の生産時間を短縮するために、データ・プロセッシング・アプリケーションの２人以上のユーザにテキスト編集で共同作業を行うことができることが

望ましい。共同作業アプリケーション用の概念的アプローチが“Kozeptionelle Ansatze fur kooperative Applikatinen”（Informationstechnik IT 4/90、第32巻、No.4、ミュンヘン、1990年８月、pp.231-240、Schnell and Sandkuhl）に述べられている。この文献は、共同作業アプリケーションを使う２人のパートナ

ーはモデルに関し同じビュー（View）を持つべきであり、相手のパートナーのアクションにグラフィック上で追従することができるようにすべきであるという考え方に基づいている。２人のパートナーのワークステーションは、互いにサーバー・クライアント関係を備え、ネットワークを通じてイベント、リクエスト、応答あるいはエラーの配布により対話を行っている。モデル・コンポーネント、ビュー・コンポーネントおよびコントローラ・コンポーネントからなる、スモールトーク80（Smalltalk80）で使われるモデル・ビュー・コントローラ概念を参考にして、共同作業の同期を説明する。各リクエストをコントローラ／ビューに割当て、各コントローラのすべてのリクエストを受けるモデル・コンポーネントによってそのリクエストを分析することが提案されている。コントローラはユーザ・アクションをリクエストに変換し、モデルから対応する機能を実行する。共同作業アプリケーションの各ビューに対し同期メカニズムを実装する必要がある。２人より多いユーザの共同作業を実現させるために、アプリケーションの非ローカル・オペレーションをリクエストとして含むイベント・ループを使うことが提案されている。

非同期あるいは同期（リアルタイム）の共同作業をサポートする同様のシステムが「SEPIAにおけるサポーティング・コラボラティブ・ライティングおよびハイパードキュメント（Supporting Collaborative Writing and Hyperdocuments in SEPIA）」（Haak and Wilson、Proceeding of the Conference on Computer-Supported Work、1992年11月、トロント、カナダ、pp. 138-142）に開示されている。このシステムは、コンポジット・ノードを使用して、異なる活動空間のブラウザとの緊密な対話ができる。メッセージ交換により同期化が行われる。メッセージは、ユーザに割当てられたブラウザからメッセージ・ハンドラへ送られ、共同作業、同時スクローリング、ウインドウのリサイジングおよびテレポインタ・ディスプレイを処理する。効果的な同期共同作業のため、オーディオ／ビジュアル・リンクのような別の通信チャネルを使うことが提案されている。

オブジェクト指向アプリケーションは、ユーザとの一貫性のある対話（interaction）インターフェイスを反映している必要があり、どのアプリケーションが

現在アクティブであるか、またどれ程多くの同時並行ユーザがそのアプリケーションを使っているかには関係がない。本件出願人が知っている公知文献には、いずれも、すべてのオブジェクト指向アプリケーションが統一的に機能することを可能にする革新的なハードウェアおよびソフトウェア・システム機能は記載されていない。

（原文明細書第５４頁）

モデル

モデルの仕事は、データのレポジトリになること、およびデータが修正されたとき変更通知を知らせることである。変更イベントは、ビューがインテリジェントに自身を更新できるだけの、変更に関する十分な情報をもつべきである。これには、通常、選択の変更前の旧値が含まれている。約束により、変更通知はデータが修正された後送られる。

ビュー

ビューの仕事は、モデル内のデータを表示することおよび変更通知に即時に反応することである。データを十分に迅速に更新できれば、変更通知に対する最も単純な応答の１つは、TView::InvalidateALL( )を実行することである。通常のビューでは、モデル変更通知に対する迅速な応答を念頭に置いてビューを設計する必要がある。これは、見かけほど難しくない。描画プログラムの場合は、２つのオフ・スクリーン・バッファ（off-screen buffer）を採用することが、すぐれた設計である。最初のバッファには、選択されなかったオブジェクトがすべて置かれることになる。２番目のバッファは、選択されたオブジェクトを選択されなかったオブジェクトの上に合成するために使用されることになる。そのあと、２番目のバッファはスクリーンを更新するために使用されることになる。

【手続補正書】特許法第１８４条の８

【提出日】１９９５年３月１５日

【補正内容】

請求の範囲

1．ユーザ・コマンドでアプリケーションの処理ステップを開始する手段、および前記処理ステップを行った結果に対してユーザのディスプレイ上に同じビューを生成する手段を有し、データを含む前記アプリケーションの少なくとも2人の前記ユーザのリアルタイム共同作業をサポートする装置において、

(a) 前記ユーザの第1のユーザにより第1のユーザ・コマンド・オブジェクトを発行して、前記アプリケーションの前記処理ステップのあるステップを決定する手段と、

(b) 第1のデータのユーザの選択に応じて、前記第1のユーザ・コマンド・オブジェクトに関連する前記第1のデータを特定する第1のユーザ選択オブジェクトを生成する手段と、

(c) 前記第1のユーザ・コマンド・オブジェクトおよび前記第1のユーザ選択オブジェクトを前記共同作業に関係する第2のユーザに配布する手段と、

(d) 前記共同作業に関係する前記第1および第2のユーザのために、前記アプリケーションで、前記第1のユーザ選択オブジェクトに対し前記第1のユーザ・コマンド・オブジェクトを実行する処理手段を具備することを特徴とする装置。

2．ひとりのユーザだけがコマンド・オブジェクトを発行することを許可する処理手段を含むことを特徴とする請求の範囲第1項に記載の装置。

3．2人以上のユーザから発行されたコマンド・オブジェクトを解析する処理手段を含むことを特徴とする請求の範囲第1項に記載の装置。

4．前記第1のユーザ・コマンド・オブジェクトは、テキスト・プロセッシング・オブジェクトであることを特徴とする請求の範囲第1項に記載の装置。

5．前記第1のユーザ・コマンド・オブジェクトは、グラフィックまたはイメージング・コマンド・オブジェクトであることを特徴とする請求の範囲第1項に記載の装置。

6．前記第1のユーザ・コマンド・オブジェクトは、マルチメディア、データベースまたはオペレーティング・システムのコマンドであることを特徴とする請求の範囲第1項に記載の装置。

７．前記第１のユーザ・コマンド・オブジェクトは、開始フェーズ、ゼロ以上の連続フェーズおよび終了フェーズを含む反復コマンド・オブジェクトであることを特徴とする請求の範囲第１項に記載の装置。

８．前記第１のユーザ・コマンド・オブジェクトは、ひとつのアトミック・オペレーションとして前記第１のユーザ・コマンド・オブジェクトを実行する非反復コマンド・オブジェクトであることを特徴とする請求の範囲第１項に記載の装置。

９．前記共同作業者および前記アプリケーションのデータを変更する前記共同作業者の権限のリストを保持し、権限のないユーザ・コマンド・オブジェクトをトラップし、前記権限のないユーザ・コマンド・オブジェクトの権限のないユーザへの配布を防ぐ処理手段を具備することを特徴とする請求の範囲第１項から第８項のうちの少なくともひとつに記載の装置。

10．前記第１のユーザ・コマンド・オブジェクトをユーザのシステムの特性に基づいて、異なった方法で適用する処理手段を含むことを特徴とする請求の範囲第１項から第９項のうち少なくともひとつに記載の装置。

11．ユーザ・コマンドでアプリケーションの処理ステップを開始する手段、および前記処理ステップを行った結果に対してユーザのディスプレイ上に同じビューを生成する手段を使用して、データを含む前記アプリケーションの少なくとも２人の前記ユーザのリアルタイム共同作業を行う方法において、

(a) 前記ユーザの第１ユーザにより第１のユーザ・コマンド・オブジェクトを発行して、前記アプリケーションの前記処理ステップのあるステップを決定するステップと、

(b) 第１のデータのユーザの選択に応じて前記第１のユーザ・コマンド・オブジェクトに関連する前記第１のデータを特定する第１のユーザ選択オブジェクトを生成するステップと、

(c) 前記第１のユーザ・コマンド・オブジェクトおよび前記第１のユーザ選択オブジェクトを前記共同作業に関係する第２のユーザに配布するステップと、

(d) 前記共同作業に関係する前記第１および第２のユーザのために、前記ア

プリケーションで、前記第１のユーザ選択オブジェクトに対し前記第１のユーザ・コマンド・オブジェクトを実行するステップ

を具備することを特徴とする方法。

12. ひとりのユーザだけがコマンド・オブジェクトを発行することを許可するステップを含むことを特徴とする請求の範囲第11項に記載の方法。

13. ２人以上のユーザから発行されたコマンド・オブジェクトを解析するステップを含むことを特徴とする請求の範囲第11項に記載の方法。

14. 前記第１のユーザ・コマンド・オブジェクトは、テキスト・プロセッシング・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

15. 前記第１のユーザ・コマンド・オブジェクトは、グラフィックまたはイメージング・コマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

16. 前記第１のユーザ・コマンド・オブジェクトは、マルチメディア、データベースまたはオペレーティング・システムのコマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

17. 前記第１のユーザ・コマンド・オブジェクトは、開始フェーズ、ゼロ以上の連続フェーズおよび終了フェーズを含む反復コマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

18. 前記第１のユーザ・コマンド・オブジェクトは、ひとつのアトミック・オペレーションとして前記第１のユーザ・コマンド・オブジェクトを実行する非反復コマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

19. 前記共同作業者および前記アプリケーションのデータを変更する前記共同作業者の権限のリストを保持し権限のないユーザ・コマンド・オブジェクトをトラップし、前記権限のないユーザ・コマンド・オブジェクトの権限のないユーザへの配布を防ぐステップを具備することを特徴とする請求の範囲第11項から第18項のうち少なくともひとつに記載の方法。

20. 前記第１のユーザ・コマンド・オブジェクトをユーザのシステムの特性に基

づいて、異なった方法で適用するステップを含むことを特徴とする請求の範囲第11項から第19項のうち少なくともひとつに記載の方法。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Intern. Application No PCT/US 94/00341

A. CLASSIFICATION OF SUBJECT MATTER
IPC 5 G06F15/21

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	INFORMATIONSTECHNIK IT vol. 32, no. 4 , August 1990 , MUNCHEN BR pages 231 - 240 E. SCHNELL ET AL. 'Konzeptionelle Ansätze für kooperative Applikationen' see the whole document	1-8, 11-18
X	PROCEEDINGS OF THE CONFERENCE ON COMPUTER-SUPPORTED COOPERATIVE WORK, CSCW '92 4 November 1992 , TORONTO, CANADA pages 138 - 146 J. HAAKE ET AL. 'Supporting Collaborative Writing of Hyperdocuments in SEPIA' see the whole document	1-8, 11-18
A	EP,A,0 319 232 (XEROX CORPORATION) 7 June 1989 see the whole document	1-8, 11-18

☐ Further documents are listed in the continuation of box C. ☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 June 1994	09.06.94

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Katerbau, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Interna[illegible] Application No
PCT/US 94/00341

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0319232	07-06-89	US-A-	5008853	16-04-91
		JP-A-	2002450	08-01-90
		US-A-	5220657	15-06-93

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(51) Int. Cl.[6]	識別記号	庁内整理番号	FI	
G06F 17/60		9288－5L	G06F 15/20	596B

(81)指定国 EP(AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), AT, AU, BB, BG, BR, BY, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, UZ, VN

(72)発明者 ペールヴィッチ, ジャック, エイチ.
アメリカ合衆国 94087 カリフォルニア州 サニーベイル ラーク レーン 1759
(72)発明者 ローゼンシュタイン, ラリー, エス.
アメリカ合衆国 95051 カリフォルニア州 サンタ クララ ミュアー アヴェニュ 182

【公報種別】特許法第１７条第１項及び特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１１年（１９９９）７月１３日

【公表番号】特表平８―５０９８２５
【公表日】平成８年（１９９６）１０月１５日
【年通号数】
【出願番号】特願平６―５１８９５６
【国際特許分類第６版】
G06F 17/00
3/14 310
9/46 340
13/00 351
17/21
17/60
【ＦＩ】
G06F 15/20 Z
3/14 310 E
9/46 340 A
13/00 351 F
15/21 Z
15/20 596 B

手 続 補 正 書

平成11年3月26日

特許庁長官 殿

1．事件の表示
特願平6－518956号

2．補正をする者
オブジェクト テクノロジー ライセンシング コーポレイション

3．代 理 人
東京都港区赤坂2丁目6番20号
電話 (03)3589 1201 (代表)
(7748) 弁理士 谷 義一

4．補正命令の日付
自 発

5．補正対象書類名
明 細 書

6．補正対象項目名
請求の範囲

7．補正の内容
請求の範囲を別紙のとおり補正する。

別 紙

請求の範囲

1．ユーザ・コマンドでアプリケーションの処理ステップを開始する手段、および前記処理ステップを行った結果に対してユーザのディスプレイ上に同じビューを生成する手段を有し、データを含む前記アプリケーションの少なくとも２人の前記ユーザのリアルタイム共同作業をサポートする装置において、
(a) 前記ユーザの第１のユーザにより第１のユーザ・コマンド・オブジェクトを発行して、前記アプリケーションの前記処理ステップのあるステップを決定する手段と、
(b) 第１のデータのユーザの選択に応じて、前記第１のユーザ・コマンド・オブジェクトに関連する前記第１のデータを特定する第１のユーザ選択オブジェクトを生成する手段と、
(c) 前記第１のユーザ・コマンド・オブジェクトおよび前記第１のユーザ選択オブジェクトを前記共同作業に関係する第２のユーザに配布する手段と、
(d) 前記共同作業に関係する前記第１および第２のユーザのために、前記アプリケーションで、前記第１のユーザ選択オブジェクトに対し前記第１のユーザ・コマンド・オブジェクトを実行する処理手段と
を具備することを特徴とする装置。

2．ひとりのユーザだけがコマンド・オブジェクトを発行することを許可する処理手段を含むことを特徴とする請求の範囲第１項に記載の装置。

3．２人以上のユーザから発行されたコマンド・オブジェクトを解析する処理手段を含むことを特徴とする請求の範囲第１項に記載の装置。

4．前記第１のユーザ・コマンド・オブジェクトは、テキスト・プロセッシング・オブジェクトであることを特徴とする請求の範囲第１項に記載の装置。

5. 前記第1のユーザ・コマンド・オブジェクトは、グラフィックまたはイメージング・コマンド・オブジェクトであることを特徴とする請求の範囲第1項に記載の装置。

6. 前記第1のユーザ・コマンド・オブジェクトは、マルチメディア、データベースまたはオペレーティング・システムのコマンド・オブジェクトであることを特徴とする請求の範囲第1項に記載の装置。

7. 前記第1のユーザ・コマンド・オブジェクトは、開始フェーズ、ゼロ以上の連続フェーズおよび終了フェーズを含む反復コマンド・オブジェクトであることを特徴とする請求の範囲第1項に記載の装置。

8. 前記第1のユーザ・コマンド・オブジェクトは、ひとつのアトミック・オペレーションとして前記第1のユーザ・コマンド・オブジェクトを実行する非反復コマンド・オブジェクトであることを特徴とする請求の範囲第1項に記載の装置。

9. 前記共同作業者および前記アプリケーションのデータを変更する前記共同作業者の権限のリストを保持し、権限のないユーザ・コマンド・オブジェクトをトラップし、前記権限のないユーザ・コマンド・オブジェクトの権限のないユーザへの配布を防ぐ処理手段を具備することを特徴とする請求の範囲第1項から第8項のうちの少なくともひとつに記載の装置。

10. 前記第1のユーザ・コマンド・オブジェクトをユーザのシステムの特性に基づいて、異なった方法で適用する処理手段を含むことを特徴とする請求の範囲第1項から第9項のうち少なくともひとつに記載の装置。

11. ユーザ・コマンドでアプリケーションの処理ステップを開始する手段、および前記処理ステップを行った結果に対してユーザのディスプレイ上に同じビューを生成する手段を使用して、データを含む前記アプリケーションの少なくとも2人の前記ユーザのリアルタイム共同作業を行う方法において、

(a) 前記ユーザの第1ユーザにより第1のユーザ・コマンド・オブジェクトを発行して、前記アプリケーションの前記処理ステップのあるステップを決定するステップと、

(b) 第1のデータのユーザの選択に応じて前記第1のユーザ・コマンド・オブジェクトに関連する前記第1のデータを特定する第1のユーザ選択オブジェクトを生成するステップと、

(c) 前記第1のユーザ・コマンド・オブジェクトおよび前記第1のユーザ選択オブジェクトを前記共同作業に関係する第2のユーザに配布するステップと、

(d) 前記共同作業に関係する前記第1および第2のユーザのために、前記アプリケーションで、前記第1のユーザ選択オブジェクトに対し同記第1のユーザ・コマンド・オブジェクトを実行するステップと

を具備することを特徴とする方法。

12. ひとりのユーザだけがコマンド・オブジェクトを発行することを許可するステップを含むことを特徴とする請求の範囲第11項に記載の方法。

13. 2人以上のユーザから発行されたコマンド・オブジェクトを解析するステップを含むことを特徴とする請求の範囲第11項に記載の方法。

14. 前記第1のユーザ・コマンド・オブジェクトは、テキスト・プロセッシング・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

15. 前記第1のユーザ・コマンド・オブジェクトは、グラフィックまたはイメージング・コマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

16. 前記第1のユーザ・コマンド・オブジェクトは、マルチメディア、データベースまたはオペレーティング・システムのコマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

17. 前記第1のユーザ・コマンド・オブジェクトは、開始フェーズ、ゼロ以上の連続フェーズおよび終了フェーズを含む反復コマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

18. 前記第1のユーザ・コマンド・オブジェクトは、ひとつのアトミック・オペレーションとして前記第1のユーザ・コマンド・オブジェクトを実行する非反復コマンド・オブジェクトであることを特徴とする請求の範囲第11項に記載の方法。

19. 前記共同作業者および前記アプリケーションのデータを変更する前記共同作業者の権限のリストを保持し権限のないユーザ・コマンド・オブジェクトをトラップし、前記権限のないユーザ・コマンド・オブジェクトの権限のないユーザへの配布を防ぐステップを具備することを特徴とする請求の範囲第11項から第18項のうち少なくともひとつに記載の方法。

20. 前記第1のユーザ・コマンド・オブジェクトをユーザのシステムの特性に基づいて、異なった方法で適用するステップを含むことを特徴とする請求の範囲第11項から第19項のうち少なくともひとつに記載の方法。

(以下余白)